TUGAS AKHIR - IF184802

KLASIFIKASI SINYAL P300 MENGGUNAKAN PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, DAN SUPPORT VECTOR MACHINE

VINCENTIUS
NRP 05111540000159

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - IF184802

KLASIFIKASI SINYAL P300 MENGGUNAKAN PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, DAN SUPPORT VECTOR MACHINE

VINCENTIUS
NRP 05111540000159

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

P300 SIGNAL CLASSIFICATION USING PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, AND SUPPORT VECTOR MACHINE

Vincentius
NRP 05111540000159

Supervisor I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

Supervisor II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KLASIFIKASI SINYAL P300 MENGGUNAKAN PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, DAN SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
Vincentius
NRP : 05111540000159

Disetujui oleh Dosen Pembimbing Tugas Akhir

Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. .....
NIP. 194908231976032001 (pembimbing 1)

Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
NIP. 197512202001122002 (pembimbing 2)

SURABAYA
JANUARI 2019

[Halaman ini sengaja dikosongkan]

KLASIFIKASI SINYAL P300 MENGGUNAKAN PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, DAN SUPPORT VECTOR MACHINE

Nama Mahasiswa	**: VINCENTIUS**
NRP	**: 05111540000159**
Jurusan	**: Departemen Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Dosen Pembimbing 2	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**

ABSTRAK

Brain Computer Interface merupakan salah satu metode interaksi yang dapat digunakan oleh penyandang cacat untuk dapat berinteraksi dengan komputer. Interaksi terjadi secara langsung antara otak pengguna dengan komputer melalui pengambilan sinyal electroencephalogram (EEG) yang kemudian diklasifikasi untuk mendeteksi Sinyal P300.

Pada tugas akhir ini, dilakukan eksperimen untuk mengklasifikasi Sinyal P300 dengan menerapkan variasi pemilihan channel sinyal EEG dengan Principal Component Analysis (PCA). Kemudian, untuk mempercepat proses training, dilakukan reduksi dimensi temporal dan channel lebih lanjut menggunakan Linear Discriminant Analysis (LDA). Proses klasifikasi sinyal EEG dilakukan dengan algoritma Support Vector Machine (SVM).

Uji coba dilakukan menggunakan dataset Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004. Hasil dari uji coba menghasilkan rata-rata f1-score terbaik sebesar 86% untuk subjek A dan 90% untuk subjek B. Proses Linear Discriminant Analysis (LDA) berhasil mengurangi waktu

training hingga dibawah 1 detik untuk semua skenario uji coba. Rata-rata f1-score terbaik setelah dilakukan LDA untuk subjek A adalah sebesar 84% dan untuk subjek B sebesar 89%.

Kata kunci: Brain Computer Interface; Electroencephalogram; Linear Discriminant Analysis; Principal Component Analysis; Sinyal P300; Support Vector Machine.

P300 SIGNAL CLASSIFICATION USING PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, AND SUPPORT VECTOR MACHINE

Student's Name	**: VINCENTIUS**
Student's ID	**: 05111540000159**
Department	**: Informatics Department FTIK-ITS**
First Advisor	**: Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Second Advisor	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**

ABSTRACT

Brain Computer Interface is one of the interaction method that can be used by people with physical disability to interact with computers. Interaction occurs directly between the user's brain to the computer by collecting the electroencephalogram (EEG) signals that can be classified to detect the P300 Signal.

In this final project, an experiment is done to classify P300 Signal by applying various channel selection of the EEG signal using Principal Component Analysis (PCA). To speed up training process, further dimensionality reduction is applied to reduce channels and temporal signal using Linear Discriminant Analysis (LDA). The classification process is done using the Support Vector Machine (SVM) algorithm.

The dataset used for this experiment is from Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004. From this experiment, the best f1-score obtained for subject A is 86% and for subject B is 90%. The LDA process successfully reduce training time to under 1 second for all experiment scenarios, resulting the f1-score of 84% for subject A and 89% for subject B.

Keywords: Brain Computer Interface; Electroencephalogram; Linear Discriminant Analysis; P300 Signal; Principal Component Analysis; Support Vector Machine.

KATA PENGANTAR

Puji syukur saya haturkan kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya saya menyelesaikan Tugas Akhir yang berjudul

KLASIFIKASI SINYAL P300 MENGGUNAKAN PRINCIPAL COMPONENT ANALYSIS, LINEAR DISCRIMINANT ANALYSIS, DAN SUPPORT VECTOR MACHINE

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS.

Tugas Akhir ini selesai karena tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Kedua orang tua, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
2. Ibu Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
3. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing II Tugas Akhir yang telah memberikan banyak yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.
4. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua Departemen Informatika ITS
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Departemen Informatika ITS.

6. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis menjalani perkuliahan di Departemen Informatika
7. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Departemen Informatika.
8. Teman-teman TC15, terimakasih banyak telah menemani penulis dari awal masa perkuliahan hingga masa pengerjaan Tugas Akhir.

Penulis memohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Januari 2019

Vincentius

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, Batasan masalah, tujuan, manfaat, metodologi pengerjaan, dan sistematika penulisan tugas akhir.

1.1. Latar Belakang

Berbagai cara sudah dilakukan produsen perangkat keras komputer untuk memungkinkan penyandang cacat dapat berinteraksi dengan komputer, seperti *braille keyboard* dan *pengenalan suara*. Untuk pengguna komputer yang memiliki keterbatasan fisik, *Brain-Computer Interface* merupakan salah satu metode interaksi yang dapat diterapkan. *Brain-Computer Interface* memungkinkan pengguna untuk melakukan komunikasi melalui perantara elektronik untuk mengirimkan pesan secara langsung dari otak ke sebuah komputer. *Brain-computer interface* melakukan pengamatan terhadap aktivitas elektrikal dari otak melalui sinyal yang disebut *electroencephalogram* (EEG). Rekaman sinyal EEG dapat menunjukkan pola dari aktivitas otak pengguna [1].

Sinyal P300 adalah komponen dari *event related potential* yang merupakan *electrophysiological response* dari stimulus visual yang diberikan. Salah satu cara untuk memberikan stimulus visual dari pengguna yaitu, dengan menampilkan matriks karakter dengan ukuran 6x6. Tugas dari pengguna adalah untuk memfokuskan perhatian ke sebuah karakter yang berada di salah satu baris atau kolom dari matriks. Semua baris dan kolom dari matriks dipancarkan secara sekuensial dengan jeda waktu tertentu. Sinyal EEG direkam selama intensifikasi baris atau kolom dari matriks. Dengan metode ini, pengguna dapat memberikan masukan berupa karakter ke sebuah komputer berdasarkan hasil rekaman dari sinyal EEG.

Agar sinyal EEG dapat dipahami oleh komputer sebagai masukan, maka perlu dilakukan pemrosesan sinyal, yaitu untuk

mendeteksi Sinyal P300 dari sinyal EEG. Kemudian dilakukan pengubahan sinyal P300 menjadi sinyal kontrol yang dapat dipahami oleh komputer. Sinyal kontrol yang dipahami oleh komputer bersifat biner, sehingga perlu dilakukan klasifikasi Sinyal EEG untuk membedakan sinyal yang merupakan sebuah masukan untuk komputer (*target*) dan yang bukan sebagai masukan (*non-target*).

Untuk melakukan ekstraksi fitur dari rekaman sinyal EEG, dilakukan pemrosesan sinyal dengan melakukan pemotongan sinyal yang berada diluar rentang waktu untuk menangkap sinyal P300 yang pada umumnya muncul sekitar 310 ms setelah intensifikasi baris atau kolom matriks [2]. Kemudian dilakukan *signal filtering* dengan *Butterworth Band Pass Filter* untuk menghilangkan *noise* dan *downsampling* [3]. Setelah dilakukan pemrosesan sinyal, digunakan algoritma pembelajaran (*machine learning*) untuk melakukan klasifikasi.

Dalam tugas akhir ini, penulis mengusulkan penggunaan algoritma pembelajaran *Support Vector Machine* (SVM) untuk melakukan klasifikasi data rekaman sinyal EEG yang diperoleh dari dataset II BCI Competition III (http://www.bbci.de/competition/iii/). *Algoritma Support Vector Machine* merupakan sebuah algoritma yang relatif handal dalam melakukan klasifikasi antara dua kelas. Selain itu, Support Vector Machine hanya memerlukan waktu yang relatif singkat untuk melakukan proses pembelajaran. Hal ini tentunya mendukung tingkat kelayakan penggunaan (*usability*) dari BCI. Untuk meningkatkan akurasi, penulis mengusulkan untuk melakukan pemilihan *channel* dengan menggunakan metode *Principal Component Analysis* (PCA) [4], dan *Linear Discriminant Analysis* (LDA) untuk meningkatkan *linear separability* per kelas dari data.

1.2. Rumusan Masalah

Permasalahan yang akan diselesaikan pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana melakukan pemrosesan rekaman sinyal EEG untuk ekstraksi fitur, meliputi pemotongan sinyal, aplikasi *Butterworth Band Pass Filter*, dan *downsampling*?
2. Bagaimana implementasi untuk menerapkan *preprocessing* data hasil ekstraksi fitur sinyal rekaman EEG menggunakan metode *Principal Component Analysis* dan, atau *Linear Discriminant Analysis*?
3. Bagaimana implementasi algoritma pembelajaran *Support Vector Machine* untuk melakukan klasifikasi sinyal P300?
4. Bagaimana mengevaluasi kinerja model yang dibentuk oleh algoritma pembelajaran *Support Vector Machine*?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data diambil dari "Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004".
2. Implementasi program menggunakan bahasa pemrograman *Matlab* dan *Python*.
3. Data sinyal diklasifikasikan menjadi 2 kelas, yaitu *Target* dan *Non-target*

1.4. Tujuan

Tujuan tugas akhir ini adalah membangun model klasifikasi sinyal P300 menggunakan algoritma *Support Vector Machine* untuk membedakan sinyal yang merupakan *Target* dan *Non-target*.

1.5. Manfaat

Manfaat pembuatan Tugas akhir ini adalah membantu klasifikasi Sinyal P300 untuk *P300 speller* yang digunakan penyandang disabilitas sebagai metode interaksi dengan komputer.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. Studi literatur
   Pada tahap ini dilakukan pencarian literatur berupa jurnal atau paper yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel internet, atau sumber lain yang bisa dipertanggungjawabkan.

3. Perancangan perangkat lunak
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat rancangan dasar dari keseluruhan sistem yang akan dibuat. Kemudian dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak
   Pada tahapan ini penulis mulai mengembangkan perangkat lunak yang meliputi pemrosesan sinyal, reduksi dimensi, balancing antar kelas dari data, pembuatan model klasifikasi, dan evaluasi dari model klasifikasi.

5. Pengujian dan evaluasi
   Pada tahapan ini penulis menguji performa dari model klasifikasi menggunakan data *testing* yang sudah disediakan pada dataset "Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004"

6. Penyusunan buku Tugas Akhir.
   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.
2. Bab II Tinjauan Pustaka
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

3. Bab III Perancangan Perangkat Lunak
   Bab ini berisi penjelasan mengenai desain, perancangan, dan pemodelan proses yang digunakan dalam Tugas Akhir ini yang direpresentasikan dengan diagram alir.
4. Bab IV. Implementasi
   Bab ini merupakan pembangunan aplikasi dengan *MATLAB* dan *Python* sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab terakhir yang berisi kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.
7. Daftar Pustaka
   Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.
8. Lampiran
   Merupakan bab tambahan yang berisi daftar istilah atau kode-kode sumber yang penting pada sistem.

BAB II
DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan metode yang digunakan untuk menyelesaikan permasalahan yang dibahas pada tugas akhir ini. Penjelasan ini bertujuan untuk memberikan dasar teori yang mendasari pengembangan perangkat lunak.

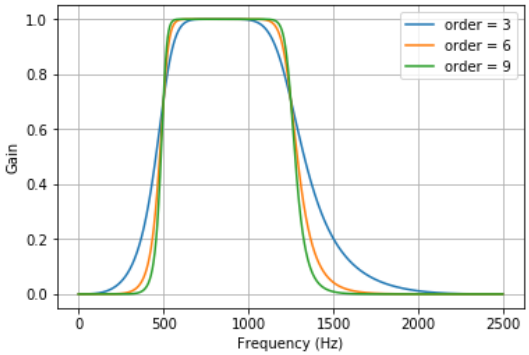
2.1. Sinyal P300

Sinyal P300 merupakan sinyal yang dihasilkan oleh otak akibat dari *event related potential* dalam bentuk *electrophysiological response* dari stimulus visual yang diberikan kepada seseorang. *Electrophysiological response* yang dihasilkan oleh seseorang berupa fluktuasi dari sinyal rekaman *electroencephalogram* EEG [1]. Sinyal P300 dapat dirangsang dengan *row/column (RC) paradigm* yang dikembangkan oleh Farwell dan Donchin. RC paradigm terdiri dari matriks dengan ukuran 6x6 yang berisi 26 karakter alfabet dan angka [1]. Tugas dari pengguna adalah memfokuskan perhatian ke karakter yang diinginkan. Tiap baris dan kolom matriks dipancarkan secara bergantian, 2 dari 12 intensifikasi baris atau kolom mengandung karakter yang diinginkan. Umumnya, fluktuasi sinyal EEG terjadi diantara 100 - 500 ms sejak stimulus diberikan, yang menandakan bahwa otak seseorang menerima stimulus. Terdapat perbedaan fluktuasi sinyal EEG dari stimulus yang mengandung karakter yang diinginkan dengan stimulus yang tidak mengandung karakter yang diinginkan. Perbedaan fluktuasi sinyal inilah yang dimanfaatkan model klasifikasi untuk membedakan karakter yang merupakan Target dan Non-target [1].

Gambar 2.1 Row/column Paradigm [1]

2.2. *Butterworth Band Pass Filter*

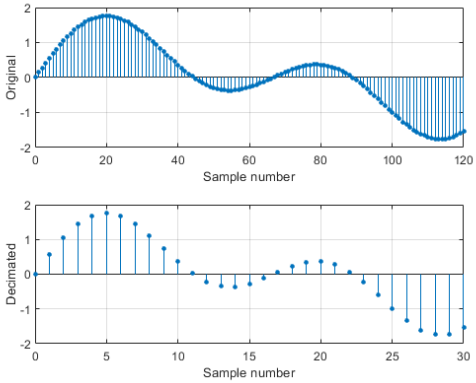
Butterworth Band Pass filter merupakan sebuah metode untuk mendesain filter analog, yang pertama kali diperkenalkan pada tahun 1930 dan masih merupakan salah satu metode yang paling banyak digunakan hingga saat ini. Tujuan dari penerapan *Butterworth Band Pass Filter* adalah mengeliminasi komponen sinyal yang berada diluar dari frekuensi jangkauan yang sudah ditentukan. Karakteristik dari *Butterworth Band Pass filter* adalah frekuensi respon yang diusahakan se-datar mungkin [5]. Untuk menghilangkan *noise* pada data EEG, pada umumnya digunakan *band pass filter* dengan frekuensi bawah 0.1 Hz dan frekuensi atas 30Hz [1]. Gambar 2.2 menunjukkan grafik respon frekuensi *Butterworth Band Pass Filter* yang akan diaplikasikan ke sinyal dengan frekuensi 5000 Hz, batas frekuensi bawah 500 Hz, batas frekuensi atas 1250 Hz, orde 3, 6 dan 9 [6]



Gambar 2.2 Grafik Respon Frekuensi *Butterworth Bandpass Filter*

2.3. *Signal Downsampling*

Sebuah operasi *signal downsampling* dengan faktor M, dimana M adalah bilangan bulat positif, dilakukan dengan cara menghapus sampel sinyal selanjutnya sebanyak M-1 secara berturut-turut dan menyimpan sampel sinyal tiap kelipatan ke-M. *Signal downsampling* dilakukan dengan tujuan untuk menghemat biaya pemrosesan sinyal [7].



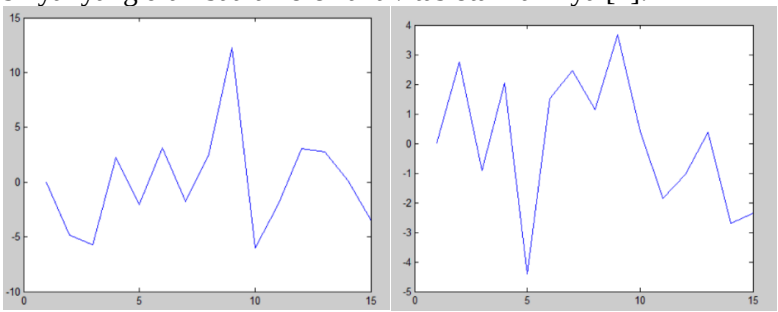
Gambar 2.3 Operasi *Signal Downsampling* dengan faktor 4 [8]

2.4. *Signal Averaging*

Signal filtering dapat bekerja dengan sangat baik untuk menghilangkan *noise* pada sinyal ketika spektrum frekuensi *noise* dan sinyal sesungguhnya tidak saling tumpang tindih. Akan tetapi seringkali spektrum frekuensi *noise* dengan sinyal sesungguhnya berada pada jangkauan yang sama sehingga aplikasi *signal filtering* akan menghilangkan karakteristik dari sinyal sesungguhnya. Oleh karena itu, dilakukan *signal averaging* dengan cara menjumlahkan beberapa rekaman sinyal dengan durasi yang sama, kemudian membanginya dengan jumlah rekaman sinyal. Dengan dilakukan *signal averaging*, *random noise* pada sinyal akan saling meniadakan sehingga memperbaiki *signal-to-noise ratio* [9]. Proses *signal averaging* didasari oleh beberapa karakteristik sinyal dan *noise* sebagai berikut [9] :

1. Gelombang sinyal harus bersifat repetitif, walaupun tidak harus bersifat periodik.
2. *Noise* harus bersifat *random* dan tidak berhubungan dengan sinyal. *Noise* yang muncul tidak periodik, sehingga hanya bisa dideskripsikan secara statistik (dengan *mean* dan varians).
3. Posisi sinyal temporal yang ingin ditangkap harus diketahui secara akurat.

Dalam pemrosesan sinyal EEG, *signal averaging* dilakukan untuk meningkatkan amplitudo sinyal P300 dan mengurangi fluktuasi sinyal yang diakibatkan oleh aktivitas otak lainnya [1].



Gambar 2.4 Sampel sinyal *electroencephalogram* (EEG) asli (kiri) dan hasil *signal averaging* dengan 5 sampel sinyal lainnya (kanan)

2.5. Normalisasi *Zero Mean*

Normalisasi *zero mean* merupakan proses standarisasi nilai dari tiap data x_i sehingga menjadi nilai baru x_i' berdasarkan sebuah nilai rata-rata $\bar{x}$ dan standar deviasi σ_x dengan rumus :

$$x_i' = \frac{x_i - \bar{x}}{\sigma_x} \quad \textbf{(2-1)}$$

Normalisasi *zero mean* mentransformasi data sehingga jika semua data x_i dijumlahkan, menghasilkan nilai 0. Dengan memanfaatkan nilai rata-rata dan standar deviasi, maka metode normalisasi ini lebih stabil terhadap *noise* dibandingkan metode min-max. Tujuan dari normalisasi adalah agar data dapat diklasifikasi dengan lebih baik dan cepat [10].

2.6. *Principal Component Analysis*

Principal Component Analysis (PCA) merupakan suatu proses proyeksi data ke basis koordinat baru yang dapat merepresentasikan varians data sebesar mungkin dengan dimensi yang sesedikit mungkin. PCA umumnya dilakukan setelah normalisasi *zero mean*. Seberapa baik basis koordinat dalam merepresentasikan varians dari dataset dipengaruhi oleh korelasi diantara atribut-atribut pada dataset, sehingga varians pada dataset jika diproyeksikan ke sistem koordinat tertentu dapat diketahui secara langsung melalui matriks kovarian dari dataset itu sendiri [11].

Sebuah dataset D dengan ukuran $n \times d$ akan memiliki matriks kovarian C dengan ukuran $d \times d$, sehingga C_{ij} merepresentasikan kovarian antara atribut (dimensi) kolom ke-i dan kolom ke-j [11]

$$C_{ij} = \frac{\sum_{k=1}^{n} x_{ik} x_{jk}}{n} - \mu_i \mu_j \qquad \forall i, j \in \{1 \dots d\} \quad \textbf{(2-2)}$$

Dimana :

- μ_i dan μ_j merepresentasikan rata-rata data di dimensi i dan j

- x_{ik} dan x_{jk} merepresentasikan data di baris ke-k, dimensi i dan j
- n merupakan jumlah baris data

Dengan rumus diatas, untuk menghitung matriks kovarian dengan dari dataset dengan jumlah dimensi d memerlukan komputasi sebesar $d \times d$, sehingga dapat disederhanakan dengan rumus

$$\boldsymbol{C = \frac{D^T D}{n} - \bar{\mu}^T \bar{\mu}} \quad \textbf{(2-3)}$$

Dimana $\bar{\mu}$ adalah vektor baris yang merepresentasikan rata-rata data untuk setiap dimensi d.

Dari matriks kovarian, dilakukan perhitungan *eigenvalue* dan *eigenvector* menggunakan persamaan karakteristik :

$$\boldsymbol{|\lambda I - C| = 0} \quad \textbf{(2-4)}$$

$$\boldsymbol{C\bar{v} - \lambda\bar{v} = 0} \quad \textbf{(2-5)}$$

Dimana λ merupakan *eigenvalue*, $\bar{v}$ merupakan *eigenvector*, dan I merupakan matriks identitas. *Eigenvector* yang dapat mempertahankan varians dari data paling baik adalah *eigenvector* yang memiliki *eigenvalue* paling tinggi. Selanjutnya dilakukan transformasi data dengan perkalian matriks antara *eigenvector* dan tiap baris data k [11].

2.7. *Linear Discriminant Analysis*

Jika PCA memproyeksikan data ke basis koordinat yang dapat merepresentasikan varians dari data sebesar mungkin, *Linear Discriminant Analysis* (LDA) memproyeksikan data ke basis koordinat yang dapat memaksimalkan pemisahan antar kelas. Agar pemisahan antar kelas data dapat dilakukan dengan baik, maka dilakukan pemilihan *eigenvector* yang memiliki *Fisher Score* paling tinggi. *Fisher Score* merupakan rasio antara sebaran data antar kelas dengan sebaran data dalam kelas yang sama. *Fisher Score* untuk proyeksi ke vektor baris $\bar{w}$ dengan dimensi sebanyak d didefinisikan dengan persamaan 2-6 [11]

$$\boldsymbol{FS(\bar{w}) = \frac{\bar{w}\, S_b\, \bar{w}^T}{\bar{w}\, S_w\, \bar{w}^T}} \quad \textbf{(2-6)}$$

$$\boldsymbol{S_b = [(\bar{\mu}_1 - \bar{\mu}_0)^T (\bar{\mu}_1 - \bar{\mu}_0)]} \quad \textbf{(2-7)}$$

$$\boldsymbol{S_w = (p_0 \Sigma_0 + p_1 \Sigma_1)} \quad \textbf{(2-8)}$$

Dimana :

- S_b merupakan *scatter matrix* antar kelas
- S_w merupakan *scatter matrix* dalam kelas yang sama
- $\bar{\mu}_1$ dan $\bar{\mu}_0$ merupakan vektor baris dengan dimensi sebanyak d yang merepresentasikan nilai rata-rata data untuk kelas 1 dan 0
- Σ_0 dan Σ_1 merupakan matriks kovarian untuk data di kelas 0 dan 1 dengan ukuran $d \times d$
- p_0 dan p_1 merupakan nilai pecahan yang merupakan perbandingan jumlah anggota antara kelas 0 dan 1

Untuk memaksimalkan *Fisher Score*, maka perlu memaksimalkan $\bar{w}\, S_b\, \bar{w}^T$ dengan asumsi $\bar{w}\, S_w\, \bar{w}^T = 1$, sehingga menyetel relaksasi Langrangian ke 0

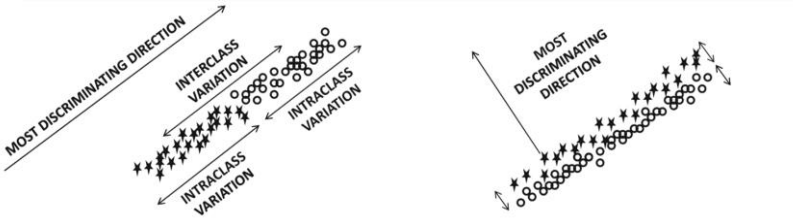
$$\boldsymbol{\bar{w}\, S_b\, \bar{w}^T - \lambda \left(\bar{w}\, S_w\, \bar{w}^T - 1\right) = 0} \quad \textbf{(2-9)}$$

menghasilkan kondisi *eigenvector* yang tergeneralisasi

$$\boldsymbol{S_b\, \bar{w}^T = \lambda\, S_w\, \bar{w}^T} \quad \textbf{(2-10)}$$

Karena $S_b\, \bar{w}^T$ selalu mengarah ke arah yang sama dengan $(\bar{\mu}_1{}^T - \bar{\mu}_0{}^T)$, maka didapatkan $\bar{w}$ optimum [11]

$$\boldsymbol{\bar{w} = (\bar{\mu}_1 - \bar{\mu}_0)\, S_w{}^{-1}} \quad \textbf{(2-11)}$$



Gambar 2.5 Kasus ketika basis koordinat dengan varians maksimum juga dapat memaksimalkan pemisahan antar kelas (kiri). Kasus ketika basis koordinat dengan varians maksimum tidak dapat memaksimalkan pemisahkan antar kelas (kanan) [11].

2.8. *Adaptive Synthetic Sampling*

Adaptive Synthetic Sampling (ADASYN) merupakan suatu algoritma yang bertujuan untuk membuat data sintetis agar kelas minor pada dataset tetap dapat diklasifikasi dengan baik. Ide dari algoritma ini adalah penggunaan bobot distribusi untuk masing-masing kelas minor berbeda yang bergantung pada tingkat kesulitan tiap kelas minor untuk diklasifikasi. Jumlah data sintetis yang akan dibuat untuk sampel data kelas minor yang sulit diklasifikasi akan semakin banyak dibandingkan sampel data kelas minor yang mudah diklasifikasi [12]. Langkah-langkah yang dilakukan untuk membuat data sintetis yaitu :

1. Menghitung derajat ketidakseimbangan antara kelas major dan kelas minor

$$\boldsymbol{d = \frac{ms}{ml}, d \in (0, 1]} \qquad \textbf{(2-12)}$$

   Dimana ms adalah jumlah data pada kelas minor dan ml adalah jumlah data pada kelas major
2. Jika $d < d_{th}$, dimana d_{th} merupakan toleransi maksimum derajat ketidakseimbangan, lakukan langkah-langkah dibawah ini :
   a. Hitung jumlah dari data sintetis yang perlu dibuat :

$$\boldsymbol{G = (ml - ms) \times \beta,\ \beta \in [0, 1]} \qquad \textbf{(2-13)}$$

Dimana β merupakan parameter yang mengatur seberapa seimbang kelas pada dataset setelah dilakukan pembuatan data sintetis. $\beta = 1$ menandakan hasil akhir dari pembuatan data sintetis adalah dataset yang seimbang dengan sempurna.

b. Untuk setiap sampel data pada kelas minoritas, cari *K nearest neighbor* berdasarkan *Euclidian distance* dalam dimensi n, dan hitung rasio seperti yang didefinisikan pada persamaan 2-14.

$$\boldsymbol{r_i = \frac{\Delta_i}{K},\ i = 1,\ \ldots., ms} \qquad \textbf{(2-14)}$$

Dimana Δ_i merupakan jumlah sampel dalam *K nearest neighbor* dari data x_i yang merupakan bagian dari kelas major. Sehingga $r_i \in [0,1]$

c. Normalisasi r_i dengan persamaan 2-15

$$\boldsymbol{\hat{r_i} = r_i / \sum_{i=1}^{ms} r_i} \qquad \textbf{(2-15)}$$

sehingga $\hat{r_i}$ adalah distribusi densitas ($\sum_i \hat{r_i} = 1$)

d. Hitung jumlah sampel data sintetis yang akan dibuat untuk setiap sampel pada kelas minoritas x_i

$$\boldsymbol{g_i = \hat{r_i} \times G} \qquad \textbf{(2-16)}$$

Dimana G merupakan jumlah data sintetis yang harus dibuat pada persamaan 2-13.

e. Selanjutnya, dilakukan pembuatan data sintetis sebanyak g_i untuk setiap data x_i dengan cara memilih salah satu data yang merupakan kelas minoritas x_{zi} pada *K nearest neighbor* untuk data x_i.

$$s_i = x_i + (x_{zi} - x_i) \times \lambda \quad \textbf{(2-17)}$$

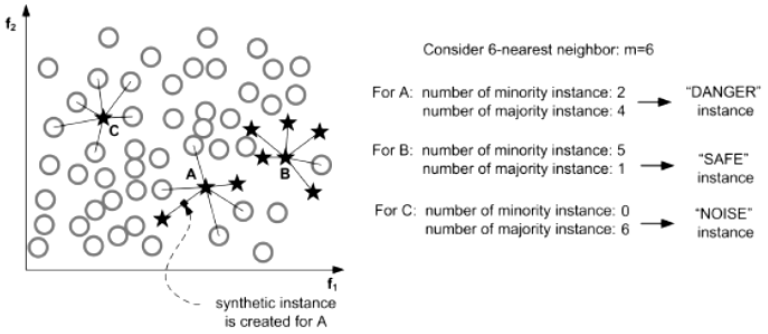
Dimana $(x_{zi} - x_i)$ merupakan jarak dalam vektor n dimensi, dan λ merupakan angka random $\lambda \in [0,1]$ [12]

2.9. *Borderline SMOTE*

Borderline SMOTE merupakan suatu algoritma untuk membuat data sintetis pada kelas minor, yang merupakan pengembangan dari algoritma *SMOTE*. Pada algoritma *SMOTE*, proses pembuatan data sintetis hanya didasari oleh pengambilan sampel data secara acak pada kelas minor, tanpa mempertimbangkan kelas sampel data yang berada disekitarnya. Hal ini menyebabkan data sintetis yang dibuat memiliki sebaran yang terlalu tinggi (overgeneralisasi), sehingga terjadi tumpang tindih antar kelas data. Sebaran data akan menjadi semakin tinggi jika terdapat data *noise*. Oleh karena itu, *Borderline SMOTE* dapat menyelesaikan masalah overgeneralisasi dengan membuat data sintetis berdasarkan algoritma berikut [13]:

1. Cari *K nearest neighbor* untuk setiap data x_i, dimana x_i merupakan anggota kelas minor.
2. Hitung jumlah data yang merupakan anggota kelas major m pada *K nearest neighbor* untuk data x_i
3. Akan dibuat data sintetis dari sampel data kelas minor x_i, jika

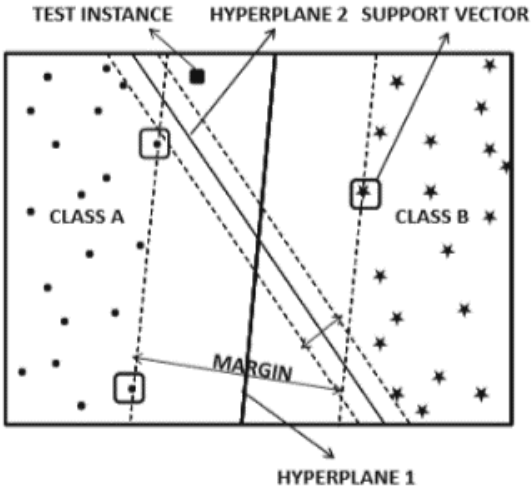
$$\frac{K}{2} \leq m < K \quad \textbf{(2-18)}$$



Gambar 2.6 Proses pemilihan sampel data kelas minor yang akan dibuat data sintetisnya, data dengan label "DANGER" yang akan dipilih [13].

2.10. *Support Vector Machine*

Support Vector Machine (SVM) merupakan suatu algoritma klasifikasi yang dibuat untuk mengklasifikasi dua kelas dari data numerik. Klasifikasi dua kelas tersebut dapat digeneralisasi sehingga SVM juga dapat digunakan untuk mengklasifikasi data dengan jumlah kelas lebih dari dua. SVM menggunakan sebuah *hyperplane* untuk memisahkan data diantara kedua kelas. Dalam kasus ini, algoritma SVM berusaha mencari sebuah *hyperplane* optimal yang memiliki jarak (*margin*) terjauh dengan data-data dari dua kelas yang posisinya terdekat dengan batas antara dua kelas tersebut (*support vectors*).



Gambar 2.7 Ilustrasi *hyperplane* yang memisahkan data antar kelas. *Hyperplane 1* dapat memisahkan data dengan lebih baik dibandingkan *hyperplane 2* karena memiliki *margin* yang lebih besar [11].

Sebuah *hyperplane* yang memisahkan data antara dua kelas didefinisikan dengan rumus sebagai berikut :

$$\overline{\boldsymbol{W}} \cdot \overline{\boldsymbol{X}} + \boldsymbol{b} = \boldsymbol{0} \quad \textbf{(2-19)}$$

Dimana $\overline{W}$ merupakan vektor baris dengan dimensi d yang merepresentasikan arah dari *hyperplane* yang sudah dinormalisasi. b merupakan *bias* yang mengatur jarak *hyperplane* dari koordinat asal (*origin*). Diasumsikan label kelas pada data yang akan diklasifikasi oleh SVM adalah $\{-1, 1\}$, sehingga :

$$\overline{\boldsymbol{W}} \cdot \overline{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \geq \boldsymbol{0}, \quad \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = +\boldsymbol{1} \quad \textbf{(2-20)}$$

$$\overline{\boldsymbol{W}} \cdot \overline{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \leq \boldsymbol{0}, \quad \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = -\boldsymbol{1} \quad \textbf{(2-21)}$$

Dimana $\bar{X}_i$ merupakan data ke i berdimensi d, dan y_i merupakan label kelas untuk data ke i.

Selain sebuah *hyperplane* yang memisahkan antar dua kelas, untuk memperkuat batasan-batasan yang mendefinisikan *margin* antar kelas, didefinisikan dua buah *hyperplane* simetris yang menyentuh *support vector* dari kedua kelas yang membutuhkan sebuaah parameter baru c yang mengatur jarak diantara kedua *hyperplane* tersebut.

$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}} + \boldsymbol{b} = +\boldsymbol{c} \quad \textbf{(2-22)}$$
$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}} + \boldsymbol{b} = -\boldsymbol{c} \quad \textbf{(2-23)}$$

Sehingga kedua *hyperplane* dapat memisahkan data ke dalam tiga daerah. Diasumsikan tidak ada data *training* yang masuk kedalam daerah diantara dua *hyperplane*. Sehingga batasan untuk data *training* dapat didefinisikan dengan :

$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \geq +\mathbf{1}, \ \ \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = +\mathbf{1} \quad \textbf{(2-24)}$$
$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \leq -\mathbf{1}, \ \ \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = -\mathbf{1} \quad \textbf{(2-25)}$$

Untuk memaksimalkan *margin*, dilakukan optimisasi untuk meminimalkan $\|\overline{\boldsymbol{W}}\|^2/2$ terhadap persamaan 2-23 dan 2-24. Masalah ini dapat diselesaikan dengan *langrangian relaxation*.

Dalam proses klasifikasi sesungguhnya, seringkali terjadi kasus dimana antara kedua kelas data tidak dapat dipisahkan dengan sempurna. Dari kasus tersebut, *hyperplane* optimal tidak dapat ditemukan. Oleh karena itu, diperlukan *soft margin*, yang memperbolehkan data *training* untuk berada pada daerah kelas yang salah dengan toleransi yang sudah ditentukan oleh *slack variable* ξ sehingga *hyperplane* pemisah antar kelas dapat didefinisikan :

$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \geq +\mathbf{1} - \boldsymbol{\xi}_{\boldsymbol{i}}, \ \ \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = +\mathbf{1} \quad \textbf{(2-26)}$$
$$\overline{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}}_{\boldsymbol{i}} + \boldsymbol{b} \leq -\mathbf{1} + \boldsymbol{\xi}_{\boldsymbol{i}}, \ \ \forall \boldsymbol{i}: \boldsymbol{y}_{\boldsymbol{i}} = -\mathbf{1} \quad \textbf{(2-27)}$$
$$\xi_i \geq 0 \quad \forall i$$

2.11. Evaluasi Model Klasifikasi

Dari model klasifikasi yang sudah dibuat dengan menggunakan data *training*, perlu dilakukan evaluasi untuk menilai seberapa baik performa dari model klasifikasi. Evaluasi ini membantu dalam proses pemilihan model klasifikasi terbaik dan juga algoritma klasifikasi yang akan digunakan. Evaluasi model klasifikasi dapat dinilai dengan *recall*, *precision,* dan *f1 score*. Ketiga metode evaluasi tersebut dapat mengevaluasi performa model dalam mengklasifikasi masing-masing kelas yang ada pada data [10].

1. Recall merupakan persentase jumlah data positif yang berhasil diklasifikasi sebagai data positif

$$\boldsymbol{recall} = \frac{\boldsymbol{TP}}{\boldsymbol{FP+FN}} \quad \textbf{(2-28)}$$

2. Precision merupakan persentase jumlah data yang diklasifikasi sebagai data positif adalah positif pada kenyataannya.

$$\boldsymbol{precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad \textbf{(2-29)}$$

3. F1-score merupakan gabungan nilai precision dan recall yang direpresentasikan dengan rata-rata harmonik dari kedua nilai tersebut.

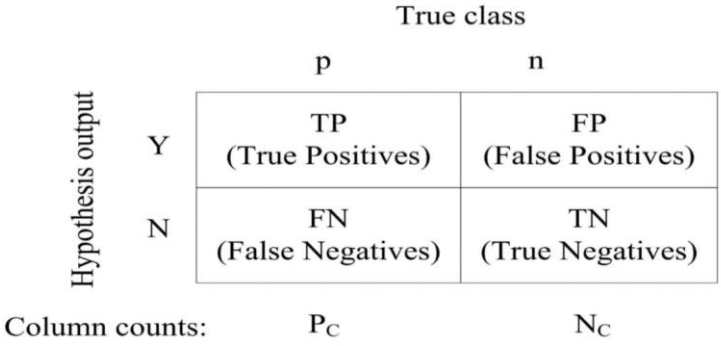
$$\boldsymbol{f1-score} = \frac{\boldsymbol{2\times precision\times recall}}{\boldsymbol{precision+recall}} \quad \textbf{(2-30)}$$

Dimana:

- TP merupakan *true positive*, yaitu jumlah data positif yang dilabeli sebagai positif (terklasifikasi dengan benar).
- TN merupakan *true negative*, yaitu jumlah data negatif yang dilabeli sebagai negatif (terklasifikasi dengan benar).
- FP merupakan *false positive*, yaitu jumlah data negatif yang dilabeli sebagai positif (terjadi kesalahan klasifikasi).

- *FN* merupakan *false negative*, yaitu jumlah data positif yang dilabeli sebagai negatif (terjadi kesalahan klasifikasi).

Keempat istilah diatas dapat direpresentasikan dalam *confusion matrix* seperti pada Gambar 2.8



Gambar 2.8 *Confusion matrix* [13]

[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan diimplementasikan. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan Tugas Akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi *pre-processing* sinyal, reduksi dimensi, *balancing* antar kelas, perancangan metode klasifikasi, dan perancangan metode evaluasi kinerja.

3.1. Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

Data yang digunakan pada proses klasifikasi sinyal P300 dibagi menjadi tiga macam yaitu data masukan, data proses dan data keluaran. Data masukan merupakan input dari pengguna perangkat lunak. Data proses adalah data sinyal EEG yang sudah diproses dan siap untuk diklasifikasi oleh algoritma klasifikasi. Sedangkan data keluaran adalah data yang ditampilkan dari hasil proses klasifikasi sinyal P300. Penjelasan masing – masing jenis data tersebut diberikan sebagai berikut.

3.1.1. Data Masukan

Data masukan adalah data yang digunakan sebagai masukan dari sistem. Data yang digunakan berupa rekaman sinyal EEG yang bersumber dari Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004 seperti yang telah dibahas di batasan masalahan tugas akhir pada subbab 1.3. Dataset ini berisi rekaman sinyal EEG dari dua subjek A dan B, yang

direkam menggunakan BCI2000 dengan paradigma *speller* yang dikembangkan oleh Donchin et al., 2000 [2]. Paradigma *speller* yang digunakan berupa matriks karakter dengan ukuran 6x6 yang ditampilkan kepada pengguna. Tugas dari pengguna adalah memfokuskan perhatian ke sebuah karakter pada matriks hingga terdapat perintah untuk memfokuskan perhatian ke karakter selanjutnya.

Dalam perekaman sinyal EEG, setiap baris dan kolom pada matriks 6x6 dipancarkan secara sekuensial dan acak, sehingga terdapat 12 kali intensifikasi baris dan kolom dalam satu putaran intensifikasi. Sinyal EEG direkam menggunakan 64 elektroda dengan frekuensi 240 Hz. Setiap baris/kolom pada matriks dipancarkan selama 100 ms, kemudian mati selama 75 ms sebelum berpindah ke baris/kolom selanjutnya. Dua dari 12 intensifikasi baris/kolom merupakan karakter yang merupakan *target*. Dataset disajikan dalam format variabel *matlab*.

Terdapat 85 karakter *target* pada data *training* dan 100 karakter *target* pada data testing. Untuk setiap karakter *target*, dilakukan pengambilan sinyal dengan melakukan 15 kali putaran intensifikasi, sehingga dalam satu karakter terdapat 180 kali intensifikasi baris/kolom. Data rekaman sinyal disimpan di dalam matriks berukuran 85x7794x64 untuk data *training* dan matriks berukuran 100x7794x64 untuk data *testing*. Dimensi pertama pada matriks merepresentasikan tiap karakter *target*, dimensi kedua merepresentasikan dimensi temporal untuk rekaman sinyal, dan dimensi ketiga merepresentasikan *channel* elektroda. Untuk setiap subjek, diberikan data *training* berupa kumpulan variabel *matlab* seperti pada Tabel 3.1, dengan dimensi seperti yang dijelaskan pada Tabel 3.2. Untuk data *testing*, diberikan kumpulan variabel *matlab* seperti pada Tabel 3.1 dan dimensi pada Tabel 3.2 tanpa variabel *StimulusType* yang merepresentasikan label kelas.

Tabel 3.1 Penjelasan Isi Tiap Variabel pada Dataset

No	Nama Variabel	Keterangan
1	*Signal*	Rekaman sinyal EEG.

2	*Flashing*	Bernilai 1 ketika ada baris/kolom yang sedang diintensifikasi, 0 sebaliknya.
3	*StimulusCode*	Menunjukkan nomor baris/kolom yang sedang diintensifikasi seperti pada Gambar 3.1. Bernilai 0 ketika tidak ada baris/kolom yang sedang diintensifikasi.
4	*StimulusType*	Bernilai 1 ketika baris/kolom yang sedang diintensifikasi mengandung karakter *target*. Bernilai 0 sebaliknya.
5	*TargetChar*	Karakter yang merupakan *target*.

Tabel 3.2 Penjelasan Tiap Dimensi Data pada Variabel

No	Nama Variabel	Dimensi 1	Dimensi 2	Dimensi 3
1	*Signal*	Karakter *target*	Sampel sinyal	Channel
2	*Flashing*	Karakter *target*	Sampel sinyal	Tidak ada
3	*StimulusCode*	Karakter *target*	Sampel sinyal	Tidak ada
4	*StimulusType*	Karakter *target*	Sampel sinyal	Tidak ada
5	*TargetChar*	Karakter *target*	Tidak ada	Tidak ada

Gambar 3.1 Skema Penomoran Baris/Kolom pada Matriks *Speller*

3.1.2. Data Proses

Data proses adalah data yang terbentuk hasil pemrosesan yang akan digunakan sebagai masukan dari proses selanjutnya, dan kemudian akhirnya digunakan untuk proses *training* dan *testing* pada *model* klasifikasi. Setiap intensifikasi baris/kolom dianggap sebagai satu rekaman data yang akan diklasifikasi. Untuk data *training*, terdapat 15.300 data rekaman sinyal EEG yang terbagi menjadi 2.550 data pada kelas *target* dan 12.750 data kelas *non-target*. Pada data testing, terdapat 18.000 data rekaman sinyal EEG yang terbagi menjadi 3.000 data pada kelas *target* dan 15.000 data pada kelas *non-target*. Untuk setiap subjek, terdapat data proses seperti pada Tabel 3.3 untuk masing-masing data *training* dan *testing*.

Tabel 3.3 Data Proses

No.	Nama data	Format	Keterangan
1	Sinyal hasil pemotongan	Variabel *matlab* (.mat), 3 dimensi	Merupakan keluaran dari pemotongan sinyal mentah antara 0-667 ms setelah intensifikasi.

2	Label untuk sinyal hasil pemotongan	Variabel *matlab* (.mat), 1 dimensi	Merupakan label kelas (*target*/*non-target*) dari tiap potongan sinyal
3	Sinyal setelah *filtering*	Variabel *matlab* (.mat), 3 dimensi	Sinyal setelah dilakukan *Butterworth Bandpass Filter*.
4	Sinyal setelah *downsampling*	Variabel *matlab* (.mat), 3 dimensi	Sinyal setelah dilakukan *downsampling* menjadi 15 sampel per intensifikasi.
5	Sinyal setelah *averaging*	Variabel *matlab* (.mat), 3 dimensi	Sinyal setelah dilakukan *averaging* dengan sampel sinyal lainnya pada kelas yang sama.
6	Label untuk sinyal setelah *averaging*	Variabel *matlab* (.mat), 1 dimensi	Label untuk tiap potongan sinyal setelah dilakukan *averaging*
7	Data setelah *PCA*	*Comma Separated Value* (.csv), 2 dimensi	Data setelah dilakukan reduksi dimensi menggunakan PCA.
8	Data setelah LDA	*Comma Separated Value* (.csv), 2 dimensi	Data setelah dilakukan transformasi LDA.

3.1.3. Data Keluaran

Data keluaran dari sistem ini adalah hasil dari proses-proses yang sudah dilakukan. Data yang sudah melewati *preprocessing* akan diklasifikasi menggunakan algoritma *Support Vector Machine*. Hasil dari proses klasifikasi tersebut adalah prediksi kelas pada masing-masing data *testing* dan nilai

evaluasinya, yaitu berupa *precision, recall* dan *f1-score* masing-masing kelas.

3.2. Desain Sistem

Secara garis besar ada enam tahapan utama dalam melakukan klasifikasi sinyal P300 pada tugas akhir ini, yaitu:

1. *Preprocessing* sinyal
2. Pembentukan label data
3. Reduksi dimensi
4. *Balancing* antar kelas
5. Pembuatan model klasifikasi (*training*)
6. Evaluasi kinerja (*testing*)

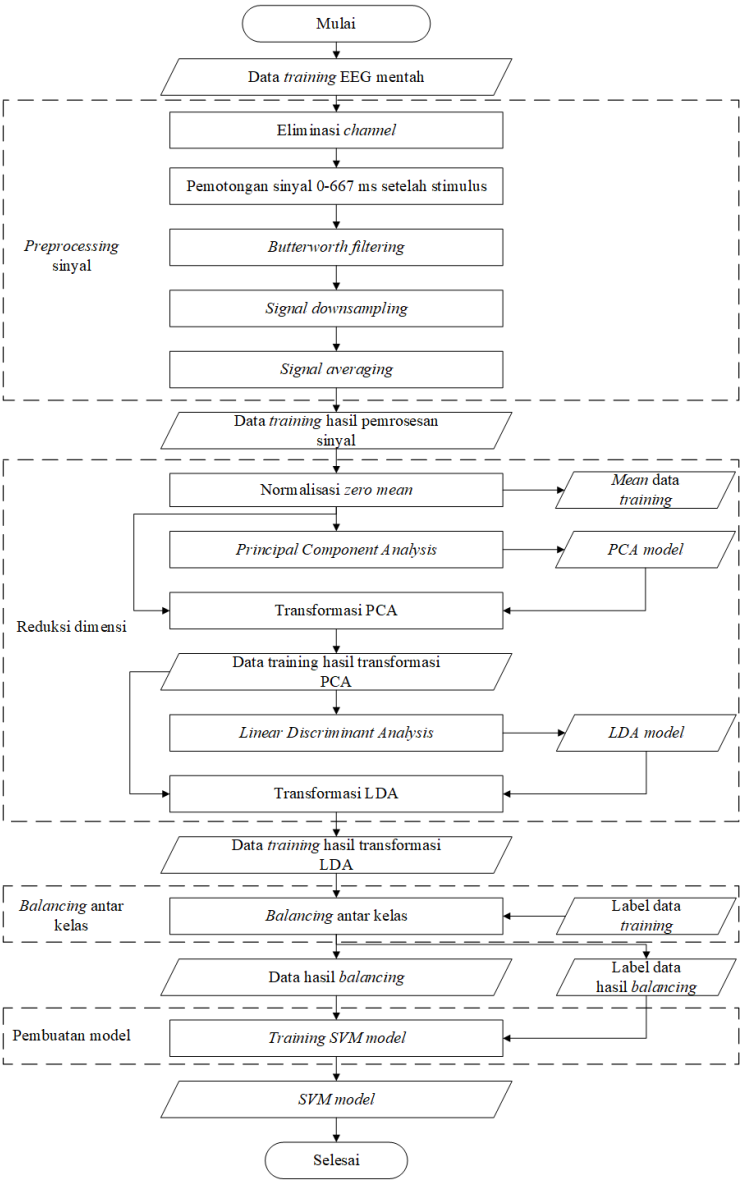
Tahapan yang pertama merupakan tahap *preprocessing* sinyal. Data yang dijadikan masukan dalam tahap *preprocessing* sinyal merupakan variabel *signal* pada data *training* dan *testing* kedua subjek. *Preprocessing* sinyal bertujuan untuk mengolah sinyal agar dapat diklasifikasi oleh algoritma klasifikasi, meliputi eliminasi *channel* untuk penghapusan *noise* spasial pada sinyal hingga proses *averaging* untuk meningkatkan *signal-to-noise ratio* [1].

Label pada data *training* dan *testing* direpresentasikan sebagai sinyal diskrit pada dimensi temporal pada data sinyal. Pada data *training*, diberikan label berupa penanda *target* atau *non-target* untuk setiap sampel sinyal (variabel *StimulusType*), dan pada data *testing* diberikan kode baris/kolom matriks yang sedang diintensifikasi untuk setiap sampel sinyal (variabel *StimulusCode*). Oleh karena itu, perlu dilakukan ekstraksi label dari data tersebut.

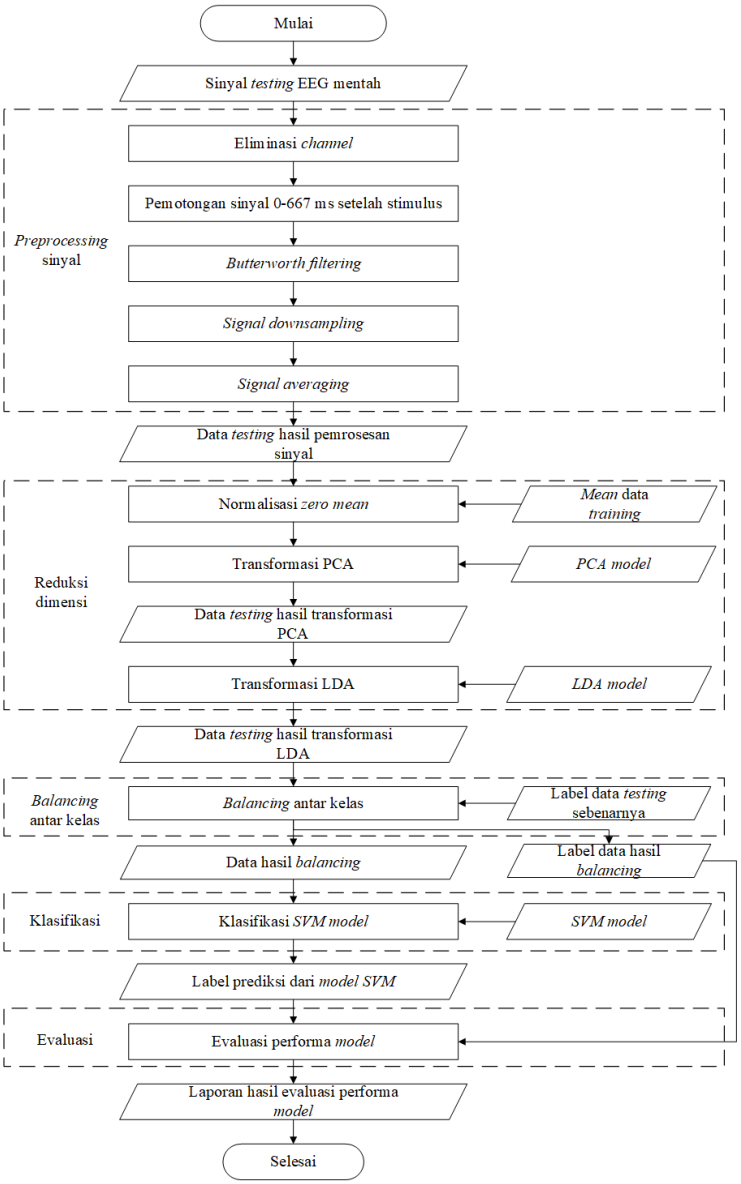
Setelah dilakukan *preprocessing* sinyal, dilakukan reduksi dimensi untuk meningkatkann performa klasifikasi. Metode reduksi dimensi yang digunakan adalah *Principal Component Analysis* (PCA), kemudian *Linear Discriminant Analysis* (LDA). Selain meningkatkan performa klasifikasi, diharapkan proses reduksi dimensi juga dapat mempercepat proses klasifikasi [11].

Jumlah data dari kelas *target* pada data sinyal lebih sedikit dibandingkan data kelas *non-target* dengan perbandingan 1:5. Hal ini disebabkan karena dari 12 baris/kolom pada matriks, hanya 2 diantaranya yang merupakan baris/kolom dari karakter target. Oleh karena itu, diperlukan proses *balancing* dataset agar *model* yang terbentuk dapat mengklasifikasi data dari kedua kelas dengan baik.

Keluaran dari proses *balancing* antar kelas merupakan data yang sudah siap untuk diklasifikasi, sehingga menjadi masukan untuk algoritma *Support Vector Machine* untuk melakukan pembentukan *model* klasifikasi. Model klasifikasi yang dibentuk akan dievaluasi performanya dengan memasukkan data *testing* untuk dilakukan prediksi kelas dari masing-masing data tersebut. Seluruh tahapan ini dilakukan untuk mengolah data yang berasal dari kedua subjek. Diagram alir untuk proses-proses yang dilakukan untuk data *training* dan *testing* dapat dilihat pada Gambar 3.2 dan 3.3



Gambar 3.2 Diagram alir untuk proses *training*



Gambar 3.3 Diagram alir untuk proses *testing*

3.2.1. Desain *Preprocessing* Sinyal

Preprocessing sinyal terdiri dari eliminasi *channel*, pemotongan sinyal EEG diantara 0-667 ms setelah intensifikasi baris/kolom, *Butterworth Band Pass Filter*, *signal downsampling*, dan *signal averaging yang* akan dijelaskan sebagai berikut.

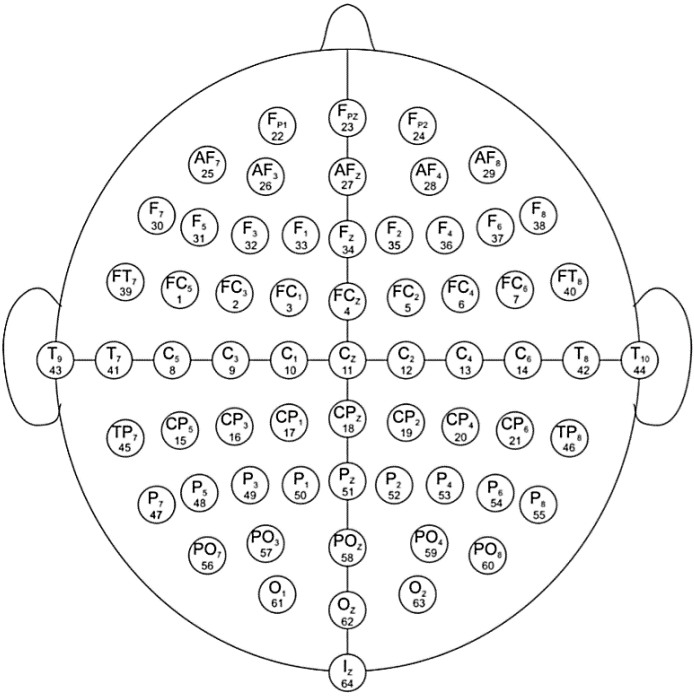
3.2.1.1. Eliminasi *Channel*

Proses eliminasi *channel* dilakukan dengan cara menghapus beberapa data pada dimensi ke-3 pada variabel *Signal*, sesuai dengan percobaan yang sudah dilakukan oleh Rakotomamonjy et al., 2008 [14]. Pada percobaan tersebut, data sinyal untuk setiap subjek dipecah menjadi 5 partisi, dan dilakukan klasifikasi menggunakan tiap partisi data. Dilakukan percobaan-percobaan untuk mengeliminasi *channel*, sehingga didapat 12 *channel* yang dapat merepresentasikan sinyal P300 dan non-P300 paling baik.

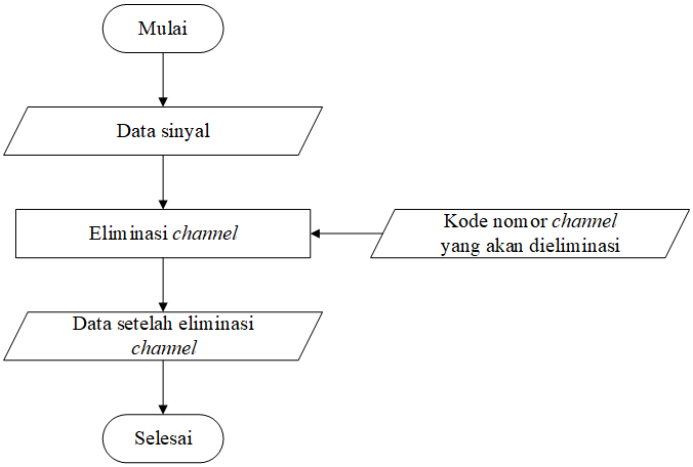
Dari data tersebut diperoleh 13 *channel* yang tidak pernah masuk kedalam 12 *channel* terbaik. 13 *channel* tersebut, yaitu FC_4, FC_6, F_{PZ}, F_{P2}, AF_3, AF_4, AF_8, F_5, FT_7, T_8, TP_8, O_Z, O_2. Dari proses eliminasi tersebut, jumlah *channel* rekaman sinyal yang akan diproses berkurang dari 64 menjadi 51. Diagram alir untuk proses penghapusan ini dapat dilihat pada Gambar 3.6. Posisi dan kode nomor setiap *channel* dapat dilihat pada Gambar 3.5.

Data	12 Top Ranked Channels											
A1	FC_1	C_2	CP_3	CP_z	F_z	F_4	F_6	P_5	P_z	P_8	PO_7	PO_8
A2	C_1	CP_z	CP_4	AF_7	AF_z	F_z	F_8	P_5	P_z	PO_7	PO_z	PO_8
A3	FC_2	CP_5	CP_1	F_1	F_z	FT_8	T_7	P_7	P_5	P_z	PO_7	PO_8
A4	C_3	C_1	F_{P1}	F_2	F_4	F_6	TP_7	P_7	P_5	P_z	PO_7	PO_8
A5	C_z	CP_5	CP_2	F_7	F_8	P_7	P_z	P_4	P_8	PO_7	PO_4	PO_8
B1	FC_5	C_5	C_z	CP_z	CP_6	AF_z	T_9	P_1	P_2	PO_7	PO_8	O_1
B2	FC_1	C_3	C_1	C_z	C_4	CP_3	CP_z	CP_4	T_9	P_1	PO_8	O_1
B3	C_1	CP_z	AF_z	T_7	T_9	P_2	P_6	PO_7	PO_z	PO_8	O_1	I_z
B4	FC_3	FC_2	CP_5	F_3	T_9	P_7	P_2	PO_7	PO_3	PO_z	PO_8	O_1
B5	FC_2	C_6	CP_z	CP_4	CP_6	T_10	P_3	P_4	PO_7	PO_8	O_1	I_z

Gambar 3.4 12 *channel* (elektroda) sinyal EEG teratas yang dapat merepresentasikan sinyal P300 dan non P300 [14].



Gambar 3.5 Posisi elektroda dan kode nomor setiap *channel*



Gambar 3.6 Diagram alir proses eliminasi *channel*

3.2.1.2. Pemotongan Sinyal EEG

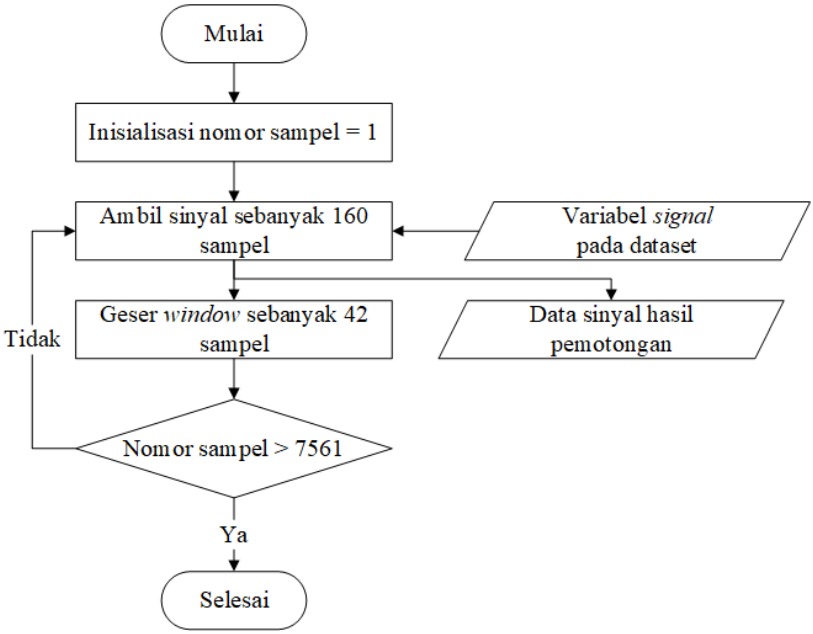
Sinyal P300 umumnya muncul dalam interval waktu 0-667 ms setelah dimulainya intensifikasi setiap baris/kolom pada matriks. Oleh karena itu, untuk setiap terjadinya intensifikasi baris/kolom, diambil rekaman sinyal sepanjang 667 ms [14]. Proses perekaman sinyal dilakukan dengan frekuensi 240 Hz, sehingga banyaknya sampel sinyal yang akan diambil dihitung dengan rumus berikut :

$$\boldsymbol{jumlah\ sampel} = \frac{\mathbf{667}}{\mathbf{1000}} \times \mathbf{240} \approx \mathbf{160}$$

Baris/kolom matriks akan diintensifikasi selama 100 ms, kemudian mati selama 75 ms sebelum berpindah ke baris/kolom selanjutnya. Sehingga satu kali siklus intensifikasi memakan waktu selama 175 ms. Jarak sampel sinyal antar intensifikasi dapat dihitung dengan rumus berikut :

$$\boldsymbol{jarak\ sampel = \frac{175}{1000} \times 240 = 42}$$

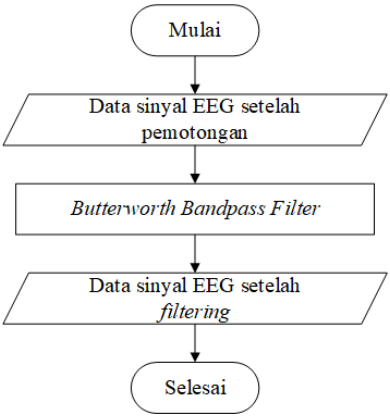
Sehingga proses pemotongan sinyal dapat dilakukan dengan membuat *sliding window* dengan ukuran 160 dan pergerakan sebesar 42 untuk setiap *channel*. Terdapat rekaman sinyal yang bersifat kontinyu untuk setiap karakter *target*, dimana untuk setiap karakter *target* terjadi intensifikasi baris/kolom matriks secara acak sebanyak 180 kali. Dari data tersebut, maka indeks awal potongan sinyal terakhir di sampel sinyal ke (42*179) +1 = 7519 untuk setiap karakter *target*. Diagram alir pemotongan sinyal dapat dilihat pada Gambar 3.7.



Gambar 3.7 Diagram alir proses pemotongan sinyal

3.2.1.3. *Butterworth Bandpass Filter*

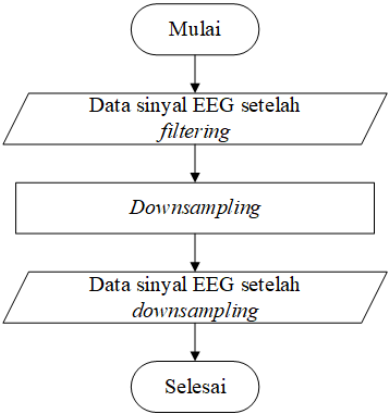
Pada rekaman sinyal EEG, terdapat *random noise* berfrekuensi tinggi yang disebabkan aktivitas lain dari otak [1]. Hal ini dapat mengurangi efektivitas klasifikasi sinyal P300. Oleh karena itu, perlu dilakukan *filtering noise* tersebut. Pada tugas akhir ini, dilakukan *filtering* dengan *Butterworth Bandpass Filter* orde 8 antara 0.1 – 20 Hz, seperti yang dilakukan pada Tugas Akhir oleh Purnomo et al., 2018 [15]. Diagram alir proses *Butterworth Bandpass Filter* untuk data *training* dan *testing* dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram alir proses *Butterworth Bandpass Filter*

3.2.1.4. Desain *Signal Downsampling*

Setelah proses *filtering*, dilakukan proses *downsampling* agar durasi proses *training* data lebih cepat. Pada Tugas Akhir ini, dilakukan *downsampling* rekaman sinyal EEG untuk setiap intensifikasi dengan faktor 11. Hasil dari proses *downsampling* ini adalah sampel sinyal yang berkurang dari 160 sampel sinyal ke 15 sampel sinyal per intensifikasi. Diagram alir proses *signal downsampling* dapat dilihat di Gambar 3.9.

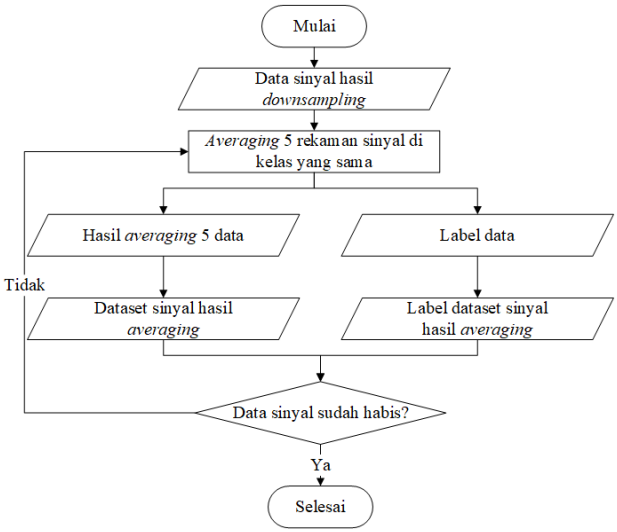


Gambar 3.9 Diagram alir proses *signal downsampling*

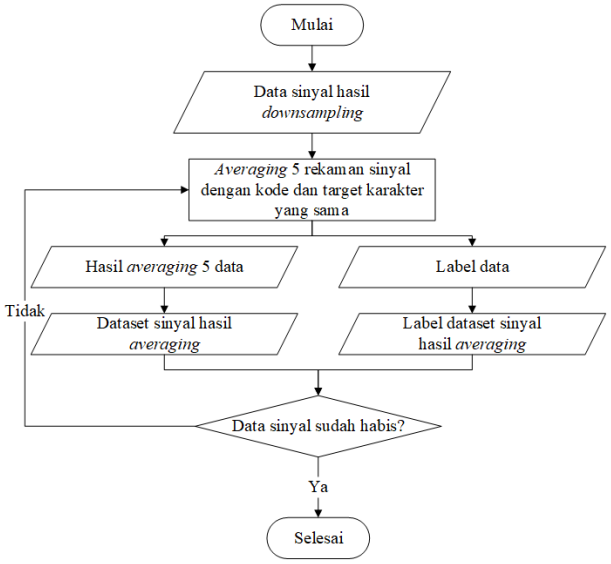
3.2.1.5. Desain *Signal Averaging*

Proses *Butterworth Bandpass filter* sudah menghilangkan *random noise* yang berada diluar dari frekuensi *filter* yang sudah ditetapkan. Kemudian, untuk menghilangkan *noise* yang memiliki frekuensi beririsan dengan sinyal P300, dilakukan *signal averaging* dengan faktor 5. Untuk data *training*, *signal averaging* dilakukan dengan cara mengakumulasi 5 rekaman sinyal EEG dengan kelas yang sama, kemudian membaginya dengan 5. Untuk data *testing*, label kelas belum diketahui, sehingga *averaging* dilakukan dengan sinyal yang berada pada intensifikasi baris/kolom matriks yang sama untuk setiap karakter target.

Proses *signal averaging* mengakibatkan penurunan jumlah rekaman sinyal EEG dengan rasio 1/5. Pada data *training*, jumlah rekaman sinyal berkurang dari 15.300 rekaman menjadi 3.060 rekaman. Pada data *testing*, jumlah rekaman sinyal berukuran dari 18.000 rekaman menjadi 3.600 rekaman. Detail proses *signal averaging* pada data *training* dan *testing* dapat dilihat pada Gambar 3.10 dan 3.11.



Gambar 3.10 Diagram alir proses *signal averaging* pada data *training*



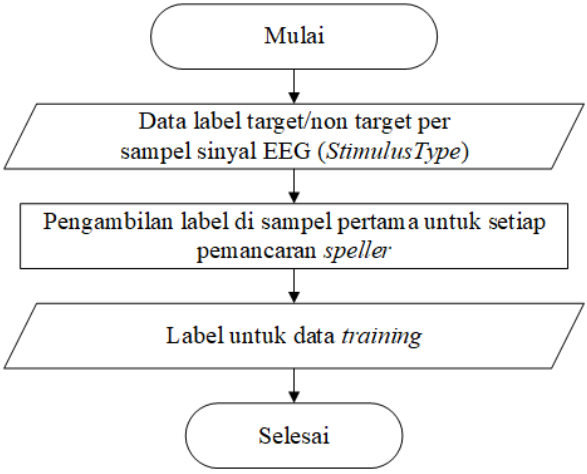
Gambar 3.11 Diagram alir proses *signal averaging* pada data *testing*

3.2.2. Desain Pembentukan Label Data

Label data yang diberikan pada dataset, yaitu variabel *StimulusType* pada data *training* dan *StimulusCode* pada data *testing* memiliki format seperti pada Tabel 3.2. Oleh karena itu, diperlukan pembentukan label data seiring dengan dilakukannya pemotongan sinyal EEG.

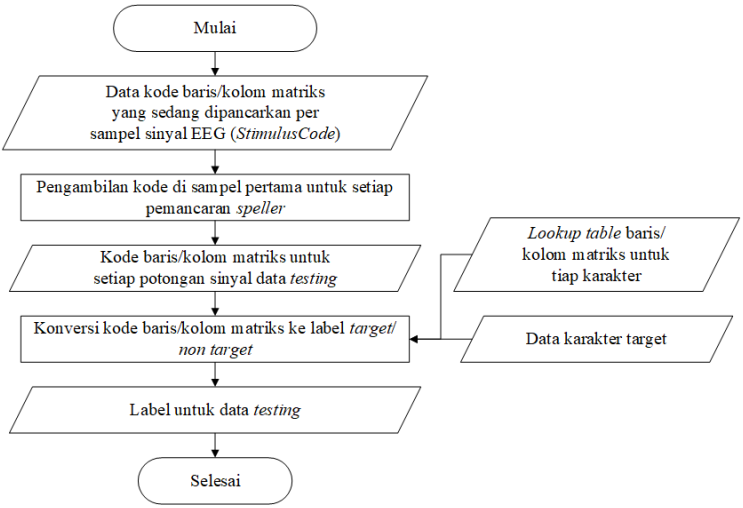
3.2.2.1. Pembentukan Label Data Dari Data Sinyal

Pada data training, diberikan sebuah variabel *StimulusType* yang merepresentasikan apakah intensifikasi saat ini mengandung karakter *target* atau tidak pada setiap sampel sinyal. Sehingga untuk mendapatkan label pada data *training*, cukup dilakukan pengambilan sampel pertama untuk setiap interval sebesar 42 (satu kali intensifikasi) pada variabel *StimulusType*. Diagram alir untuk operasi pembentukan label data untuk data *training* dapat dilihat pada Gambar 3.12.



Gambar 3.12 Diagram alir pembentukan label data untuk data *training*

Pada data *testing*, tidak terdapat variabel *StimulusType* karena awalnya dataset Wadsworth BCI Dataset (P300 Evoked Potentials) merupakan dataset yang digunakan untuk kompetisi BCI Competition III Challenge 2004, sehingga hanya diberikan data karakter *target* di akhir kompetisi. Oleh karena itu, perlu dilakukan operasi pencocokan baris/kolom yang sedang diintensifikasi dengan data karakter *target* untuk mengetahui apakah baris/kolom yang sedang diintensifikasi merupakan *target* atau *non-target*. Operasi ini memanfaatkan skema penomoran baris/kolom matriks seperti pada Gambar 3.1, data karakter *target*, dan variabel *StimulusCode* yang dijelaskan di Tabel 3.1 dan 3.2. Alur proses pembentukan label data untuk data *testing* dapat dilihat di diagram alir pada Gambar 3.13.



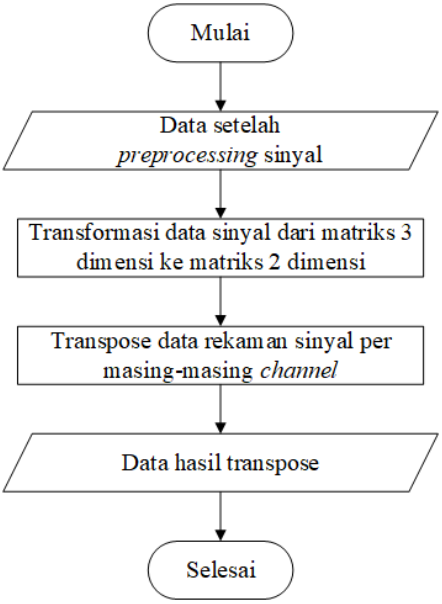
Gambar 3.13 Diagram alir pembentukan label data untuk data *testing*

3.2.2.2. Pembentukan Label Data Dari Sinyal Setelah *Averaging*

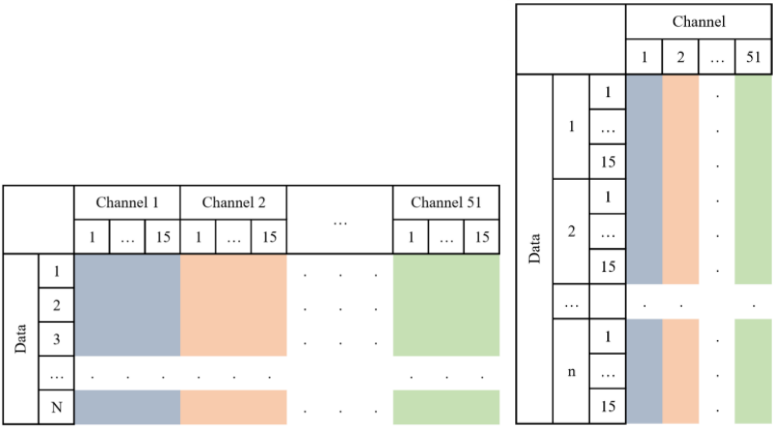
Proses *signal averaging* menggabungkan beberapa data rekaman sinyal EEG menjadi satu sehingga ukuran dan indeks label data yang dibentuk pada proses sebelumnya menjadi tidak sesuai dengan data hasil *averaging*. Label data dari data sinyal yang digabungkan juga perlu digabungkan seiring dengan proses penggabungan sinyal itu sendiri. Oleh karena itu, dalam proses *signal averaging* seperti yang dijelaskan dalam diagram alir pada Gambar 3.10 dan 3.11, label dari data sinyal hasil penggabungan juga diolah bersamaan dengan proses penggabungan itu sendiri.

3.2.3. Desain Reduksi Dimensi

Proses reduksi dimensi bertujuan untuk pemilihan *channel* yang dapat merepresentasikan sinyal P300 dengan baik menggunakan metode *Principal Component Analysis*. Setelah pemilihan channel, dilakukan reduksi pada dimensi temporal dan *channel* pada sinyal EEG menggunakan metode *Linear Discriminant* analysis. Proses *Linear Discriminant Analysis* bertujuan untuk memproyeksikan data sinyal pada dimensi *temporal* sekaligus dimensi *channel* yang dapat memisahkan kedua kelas dengan baik. Agar *Principal Component Analysis* dapat melakukan pemilihan *channel*, dilakukan transformasi data sinyal dari matriks 3 dimensi ke matriks 2 dimensi dengan cara menyambungkan data pada dimensi ke 3 ke dimensi ke 2. Kemudian rekaman sinyal dari masing-masing *channel* ditranspose sehingga menghasilkan matriks 2 dimensi dengan panjang baris 15 * 3.060 untuk data *training* dan 15 * 3.600 untuk data *testing*. Jumlah kolom pada data *training* dan *testing* merupakan jumlah *channel* pada data sinyal, yaitu sebanyak 51 kolom [16]. Setelah itu, dilakukan normalisasi *zero mean* pada data sinyal [11]. Diagram alir pemrosesan pra-reduksi dimensi dapat dilihat pada Gambar 3.14.



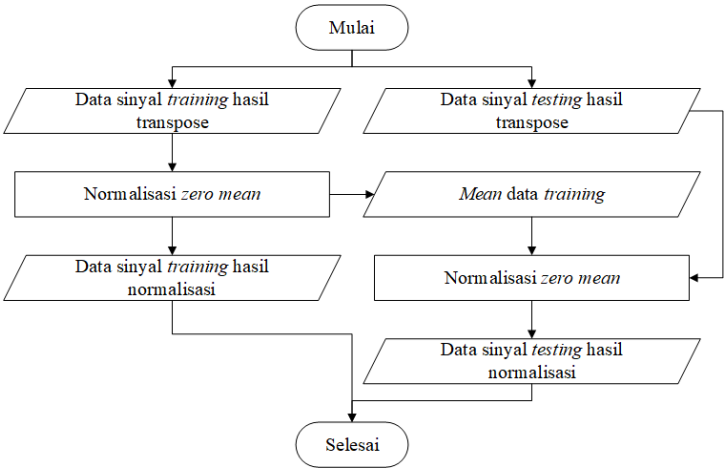
Gambar 3.14 Diagram alir pemrosesan data pra-reduksi dimensi



Gambar 3.15 Ilustrasi representasi data sinyal setelah transformasi ke matriks 2 dimensi (kiri), setelah transpose masing-masing *channel* (kanan)

3.2.3.1. Normalisasi *Zero Mean*

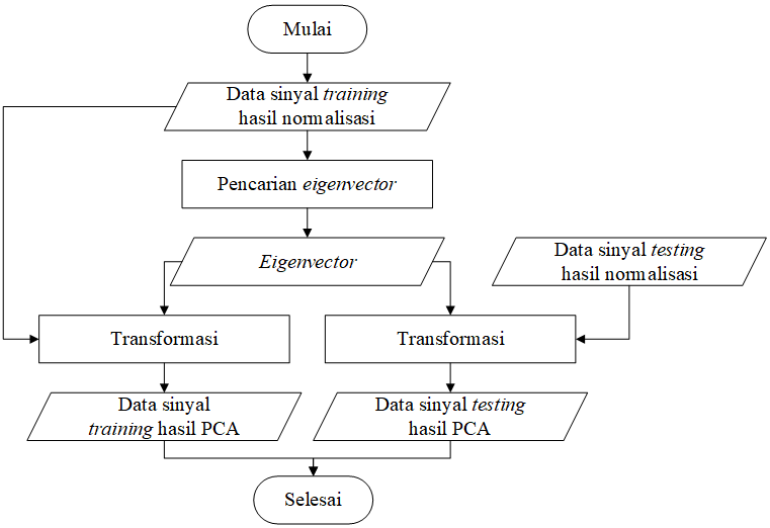
Pada data hasil transpose, dilakukan normalisasi *zero mean*, sehingga rata-rata sampel sinyal untuk setiap *channel* adalah nol. Normalisasi *zero mean* diaplikasikan pada data *training* dan *testing*. Diagram alir untuk proses normalisasi *zero mean* dapat dilihat pada Gambar 3.16.



Gambar 3.16 Diagram alir proses normalisasi *zero mean*

3.2.3.2. *Principal Component Analysis*

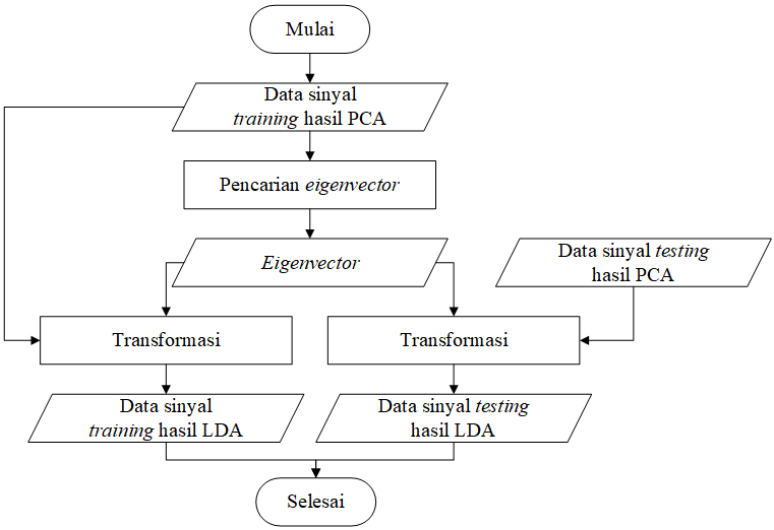
Setelah data ternormalisasi, dilakukan *Principal Component Analysis* pada dimensi spasial (*channel*). Proses pencarian *eigenvector* dilakukan menggunakan data *training*, kemudian data *training* dan *testing* ditransformasi menggunakan *eigenvector* tersebut. Diagram alir untuk proses *Principal Component Analysis* dapat dilihat pada Gambar 3.17.



Gambar 3.17 Diagram alir proses PCA

3.2.3.3. *Linear Discriminant Analysis*

Sebelum dilakukan *Linear Discriminant Analysis*, data *training* dan *testing* ditranspose kembali ke representasi semulanya, sehingga dimensi temporal pada data berada pada kolom matriks. Dilakukan pencarian *eigenvector* yang dapat mentransformasi dimensi temporal pada data sinyal ke arah yang dapat memisahkan antar kelas secara linier. Seperti pada PCA, proses pencarian *eigenvector* dilakukan menggunakan data training, yang kemudian transformasinya dilakukan ke data *training* dan *testing* menggunakan *eigenvector* tersebut. Diagram alir untuk proses *Linear Discriminant Analysis* dapat dilihat pada Gambar 3.18.



Gambar 3.18 Diagram alir proses LDA

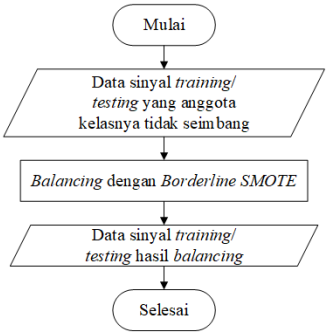
3.2.4. Desain *Balancing* Antar Kelas

Karena dalam satu kali putaran intensifikasi hanya terdapat dua baris/kolom yang merupakan target dari total 12 baris/kolom, maka perbandingan jumlah data sinyal yang termasuk kedalam kelas *target* dengan yang masuk kedalam kelas *non-target* adalah 1:5. Oleh karena itu, agar model yang terbentuk dapat mengklasifikasi kelas *target* dengan baik, maka perlu dilakukan *balancing* kelas data rekaman sinyal *training* dan *testing*. Pada tugas akhir ini, dilakukan variasi algoritma *balancing* yang dilakukan, yaitu *Borderline SMOTE*, *ADASYN*, dan duplikasi.

3.2.4.1. *Balancing* dengan *Borderline SMOTE*

Proses *balancing* dengan *Borderline SMOTE* dilakukan dengan cara memasukkan data sinyal dan label untuk tiap data sinyal sebagai masukan. Algoritma ini akan membuat data sintetis untuk kelas minor (*target*). Keluaran dari algoritma ini adalah data sinyal dan label yang jumlah anggota antar kelasnya sudah

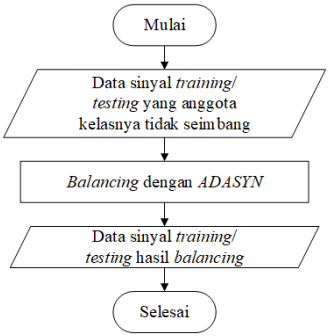
seimbang. Detail proses *balancing* dengan *Borderline SMOTE* digambarkan dengan diagram alir pada Gambar 3.19.



Gambar 3.19 Diagram alir proses *balancing* dengan *borderline SMOTE*

3.2.4.2. *Balancing* dengan *ADASYN*

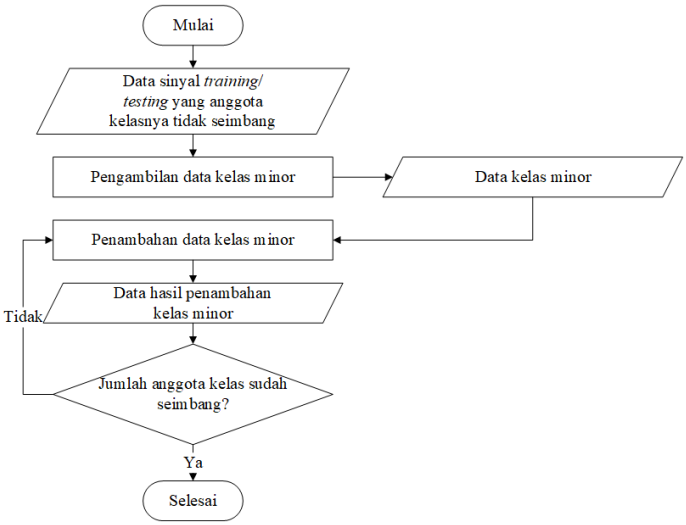
Proses balancing dengan algoritma *ADASYN* dilakukan dengan memasukkan data rekaman sinyal dan label untuk tiap data rekaman sinyal sebagai masukan. Algoritma ini akan membuat data sintetis dari data yang berada di kelas minor (*target*). Keluaran dari algoritma ini adalah data rekaman sinyal dan label yang jumlah anggota antar kelasnya sudah seimbang. Diagram alir untuk proses *balancing* dengan *ADASYN* dapat dilihat pada Gambar 3.20.



Gambar 3.20 Diagram alir proses *balancing* dengan *ADASYN*

3.2.4.3. *Balancing* dengan Duplikasi

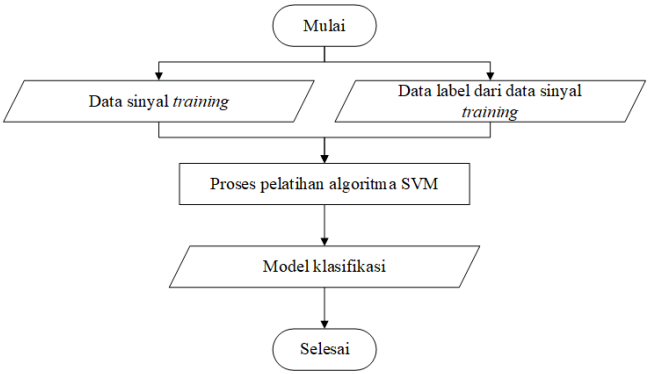
Berbeda dengan kedua algoritma sebelumnya, proses duplikasi dilakukan dengan mengumpulkan seluruh data rekaman sinyal yang termasuk kedalam kelas minor (*target*), kemudian menduplikasi data sinyal yang sudah terkumpul tersebut hingga jumlah data pada kelas minor (*target*) sama dengan jumlah data pada kelas major (*non-target*). Diagram alir untuk proses duplikasi dapat dilihat pada Gambar 3.21.



Gambar 3.21 Diagram alir proses duplikasi

3.2.5. Desain Pembuatan *Model* Klasifikasi *Support Vector Machine*

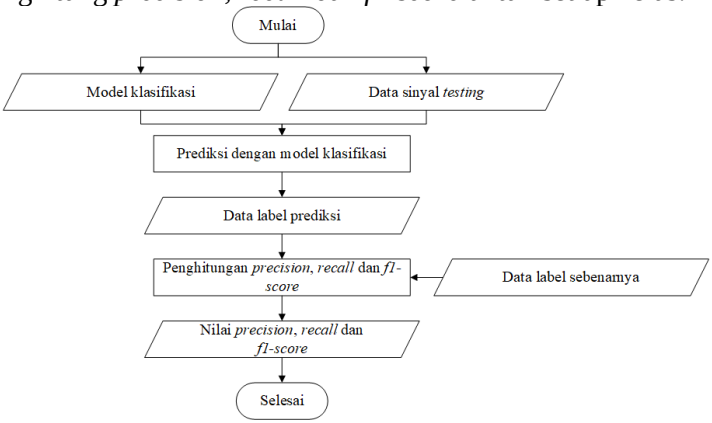
Proses pembuatan model klasifikasi pada tugas akhir ini menggunakan algoritma *Support Vector Machine* dengan *kernel linear* dan *C-parameter* 1,0. Dalam membuat model klasifikasi, masukan dari algoritma ini adalah data sinyal EEG *training* yang setidaknya sudah melalui *preprocessing* sinyal, dan label data dari masing-masing data *training*. Detail proses pembuatan *model* klasifikasi dapat dilihat pada diagram alir di Gambar 3.22.



Gambar 3.22 Diagram alir proses pembuatan *model* dengan algoritma SVM

3.2.6. Desain Evaluasi Kinerja

Untuk mengevaluasi kinerja model yang sudah dibuat, dilakukan proses klasifikasi data *testing* menggunakan *model* tersebut. Hasil klasifikasi dari data *testing* berupa label prediksi *model*, yang kemudian akan dilakukan pencocokan dengan label sebenarnya. Penilaian performa klasifikasi dilakukan dengan menghitung *precision*, *recall* dan *f1-score* untuk setiap kelas.



Gambar 3.23 Diagram alir proses evaluasi kinerja *model*

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi ini menjelaskan mengenai klasifikasi sinyal P300 dengan menampilkan kode sumber yang digunakan.

4.1. Lingkungan implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel ® Core i7-8550U CPU @ 1.80GHz (8 CPUs), ~2.0GHz Memori: 16384MB RAM
Perangkat lunak	Sistem Operasi: • Windows 10 Pro 64-bit Perangkat Pengembang: • MATLAB 7.8.0.347 (R2009a) • Python 3.6.4 • pandas 0.22.0 • scikit-learn 0.19.1 • imbalanced-learn 0.3.1 Perangkat Pembantu: • Spyder 3.3.1

4.2. Implementasi

Subbab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

4.2.1. Implementasi *Preprocessing* Sinyal

Tahap paling awal yang dilakukan pada proses klasifikasi sinyal P300 adalah melakukan *preprocessing* sinyal. Proses ini

terdiri dari eliminasi *channel*, pemotongan sinyal EEG, *Butterworth Bandpass Filter*, *signal downsampling*, dan *signal averaging*. Keluaran dari proses ini adalah dataset sinyal EEG yang dapat diklasifikasi oleh model klasifikasi menjadi kelas *target* dan *non-target*. Implementasi tahap ini melibatkan data tiga dimensi, sehingga diimplementasikan menggunakan *Matlab*.

4.2.1.1. Implementasi Eliminasi *Channel*, Pemotongan Sinyal EEG, dan *Butterworth Bandpass Filter*

Eliminasi *channel*, pemotongan sinyal EEG, dan *Butterworth Bandpass Filter* merupakan operasi yang harus berjalan secara sekuensial untuk setiap potongan sinyal. Oleh karena itu, pada kode baris ke-29 sampai 35, proses eliminasi *channel* dilakukan dengan melewati satu iterasi jika iterasi saat ini merupakan *channel* yang dieliminasi. Kemudian, untuk *channel* yang tidak dieliminasi, dilakukan pemotongan sinyal dengan kode sumber baris 36 sampai 45 sesuai dengan diagram alir pada Gambar 3.7. Dalam iterasi yang sama, dilakukan *Butterworth Bandpass Filter* pada sinyal hasil pemotongan tersebut dengan kode sumber baris ke-36. Pembuatan desain *Butterworth Bandpass Filter* dilakukan menggunakan fungsi "*butter*" pada *Matlab* yang menerima parameter orde filter, batas frekuensi atas, batas frekuensi bawah, dan frekuensi sinyal [17].

```
1.  %load dataset
2.  load('path_to_dataset')
3.
4.  %definisi channel yang dieliminasi
5.  eliminated_channels =
    [6,7,23,24,26,28,29,31,39,42,46,62,63];
6.  [r,c] = size(eliminated_channels);
7.
8.  %inisialisasi iterator dan variabel
```

```
i = 1;
j = 1;
l = 1;
m = 1;
splitted_signal = [];
current_flash = [];

%desain butterworth filter
lowFreq = 0.1;
hiFreq = 20;
fs = 240;
order = 8;
[z,p,k] = butter(order/2, [lowFreq hiFreq]/(fs/2), 'bandpass');
[sos,g] = zp2sos(z,p,k);
hd=dfilt.df2tsos(sos,g);

%pemrosesan sinyal
while i < size(Signal, 1)+1
  while j < 7542
    while l < 65
      if l == eliminated_channels(m)
        l = l+1;
        if m < c
          m = m+1;
        end
        continue;
      end
      current_channel = filter(hd, Signal(i,j:j+159,l));
      current_flash = cat(3, current_flash, current_channel);
      l = l + 1;
```

```
39.     end
40.     splitted_signal = cat(1,
    splitted_signal, current_flash);
41.     j = j + 42;
42.     l = 1;
43.     m = 1;
44.     current_flash = [];
45.   end
46.   i = i + 1;
47.   j = 1;
48. end
49.
50. %menyimpan hasil pemrosesan sinyal
51. save('destination_path', 'splitted_signal')
```

Kode Sumber 4.1 Eliminasi *Channel*, Pemotongan Sinyal EEG, dan *Butterworth Bandpass Filter*.

4.2.1.2. Implementasi *Signal Downsampling*

Pada sinyal yang sudah terpotong, terdapat 160 sampel sinyal untuk setiap data. Pada proses *signal downsampling*, dilakukan pengurangan sampel sinyal menjadi 15 sampel sinyal menggunakan fungsi "*downsample*" pada kode sumber baris ke-13 [18]. Untuk melakukan *signal downsampling* untuk setiap data, dilakukan iterasi menggunakan kode sumber baris ke-11 sampai dengan ke-21.

```
1.  %load dataset
2.  load('path_to_dataset')
3.
4.  %inisialisasi iterator dan variabel
5.  i = 1;
6.  j = 1;
7.  signal_downsampled = [];
```

```
8.  trial_downsampled = [];
9.
10. %downsampling
11. while i < size(splitted_signal, 1)+1
12.   while j<52
13.     channel_downsampled =
    downsample(splitted_signal(i,:,j),11);
14.     trial_downsampled = cat(3,
    trial_downsampled, channel_downsampled);
15.     j = j + 1;
16.   end
17.   signal_downsampled = cat(1,
    signal_downsampled, trial_downsampled);
18.   trial_downsampled = [];
19.   j = 1;
20.   i = i + 1;
21. end
22.
23. %menyimpan hasil downsampling
24. save('destination_path',
    'signal_downsampled')
```

Kode Sumber 4.2 ***Signal downsampling***

4.2.1.3. Implementasi *Signal Averaging*

Setelah proses downsampling, dilakukan proses averaging dengan menggabungkan 5 rekaman sinyal pada kelas yang sama pada data training. Proses ini dimulai dengan iterasi untuk setiap data potongan sinyal, dan memasukkan potongan sinyal ke *buffer* masing-masing kelas dengan kode sumber baris ke-17 sampai dengan 22. Setelah *buffer* berisi 5 rekaman sinyal, dilakukan penjumlahan 5 sinyal tersebut kemudian pembagian dengan faktor 5 menggunakan kode sumber baris ke-26 sampai 32 untuk *buffer* kelas *target* dan 39 sampai 45 untuk *buffer* kelas *non-target*. Setelah itu, dengan kode sumber baris ke-33

sampai 34, dilakukan penghapusan isi *buffer* kelas *target* dan dengan kode sumber baris ke-46 sampai 47 untuk *buffer* kelas *non-target*. Selain itu, dilakukan juga pembentukan label untuk data hasil *averaging* dengan kode sumber baris ke-32 dan 45.

```
1.  %load data
2.  load('path_to_dataset')
3.  load('path_to_label')
4.
5.  %inisialisasi variabel
6.  averaged_signal = [];
7.  averaged_label = [];
8.  target_sum =
    zeros(1,size(signal_downsampled,2),size(sig
    nal_downsampled,3));
9.  nontarget_sum =
    zeros(1,size(signal_downsampled,2),size(sig
    nal_downsampled,3));
10. target_buffer = [];
11. nontarget_buffer = [];
12. factor = 5;
13. i = 1;
14. j = 1;
15.
16. %averaging
17. while i < size(signal_downsampled, 1)+1
18.   if label(i) == 1
19.     target_buffer = cat(1, target_buffer,
    signal_downsampled(i,:,:));
20.   elseif label(i) == 0
21.      nontarget_buffer = cat(1,
    nontarget_buffer,
    signal_downsampled(i,:,:));
22.   end
```

```
23.
24.   %flush target buffer
25.   if size(target_buffer,1) >= factor
26.     while j <= factor
27.       target_sum = target_sum(1,:,:) +
    target_buffer(j,:,:);
28.       j=j+1;
29.     end
30.     target_sum = target_sum/factor;
31.     averaged_signal = cat(1,
    averaged_signal, target_sum);
32.     averaged_label = cat(1, averaged_label,
    1);
33.     target_buffer = [];
34.     target_sum =
    zeros(1,size(signal_downsampled,2),size(sig
    nal_downsampled,3));
35.     j=1;
36.
37.   %flush nontarget buffer
38.   elseif size(nontarget_buffer,1) >= factor
39.     while j <= factor
40.       nontarget_sum = nontarget_sum(1,:,:)
    + nontarget_buffer(j,:,:);
41.       j=j+1;
42.     end
43.     nontarget_sum = nontarget_sum/factor;
44.     averaged_signal = cat(1,
    averaged_signal, nontarget_sum);
45.     averaged_label = cat(1, averaged_label,
    0);
46.     nontarget_buffer = [];
```

```
47.     nontarget_sum =
    zeros(1,size(signal_downsampled,2),size(sig
    nal_downsampled,3));
48.     j=1;
49.   end
50.
51.   i = i + 1;
52. end
53.
54. %menyimpan hasil averaging
55. save('signal_destination_path',
    'averaged_signal')
56. save('label_destination_path ',
    'averaged_label')
```

Kode Sumber 4.3 *Signal averaging* data *training*

Pada data testing, karena label data belum diketahui sebelumnya, maka proses *signal averaging* dilakukan antara sinyal yang memiliki kode baris/kolom yang sama. Proses averaging ini dilakukan dengan membuat *buffer* sejumlah dengan baris/kolom, yaitu 12 dengan kode sumber baris ke-8. Setelah itu dilakukan pembacaan sinyal dengan iterasi dan memasukkan ke *buffer* sesuai dengan kode baris/kolomnya pada kode sumber baris ke-16 dan 17. Setelah *buffer* sudah berisi 5 potongan sinyal, maka sinyal yang terkumpul akan dibagi dengan faktor 5 dengan kode sumber baris ke-19 sampai 24.

```
1.  load('path_to_dataset')
2.  load('path_to_code')
3.  load('path_to_label')
4.
5.  %inisialisasi variabel
6.  averaged_signal = [];
```

```
7.  averaged_label = [];
8.  buffer =
    zeros(12,size(signal_downsampled,2),size(si
    gnal_downsampled,3));
9.  buffer_status = zeros(12);
10. factor = 5;
11. i = 1;
12. j = 1;
13.
14. %averaging
15. while i < size(signal_downsampled, 1) + 1
16.   buffer(code(i),:,:) = buffer(code(i),:,:)
    + signal_downsampled(i,:,:);
17.   buffer_status(code(i)) =
    buffer_status(code(i))+ 1;
18.   while j < 13
19.     if buffer_status(j) >= factor
20.       averaged_signal = cat(1,
    averaged_signal, buffer(j,:,:)/factor);
21.       averaged_label = cat(1,
    averaged_label, label(i));
22.       buffer(j,:,:) = 0;
23.       buffer_status(j) = 0;
24.     end
25.     j = j + 1;
26.   end
27.   j = 1;
28.   i = i + 1;
29. end
30.
31. %menyimpan hasil averaging
32. save('signal_destination_path',
    'averaged_signal')
```

```
33. save('label_destination_path',
    'averaged_label')
```

Kode Sumber 4.4 *Signal averaging* data *testing*

4.2.2. Implementasi Pembentukan Label Data

Implementasi pembentukan label data antara data *training* dan data *testing* dilakukan secara terpisah. Hal ini disebabkan oleh informasi yang disertakan di data *training* berbeda dengan informasi pada data *testing*.

4.2.2.1. Implementasi Pembentukan Label Data *Training*

Pada data *training*, disediakan sebuah variabel *matlab* bernama *StimulusType* seperti yang dijelaskan pada Tabel 3.1 dan 3.2 yang akan digunakan untuk membentuk label data *training*. Pengambilan label data dilakukan dengan cara iterasi di kode baris ke-4 sampai 11 untuk mengambil data pertama pada setiap intensifikasi yang berjarak 42 sampel. Karena proses pembentukan label data *training* berasal dari variabel *matlab*, maka implementasi proses ini dilakukan menggunakan *matlab*.

```
1. i = 1;
2. j = 1;
3. label = [];
4. while i < size(StimulusType, 1)+1
5.   while j < 7542
6.     label = cat(1, label,
   StimulusType(i,j));
7.       j = j + 42;
8.   end
9.   i = i + 1;
10.   j = 1;
11. end
```

Kode Sumber 4.5 Pembentukan label data *training*

4.2.2.2. Implementasi Pembentukan Label Data *Testing*

Untuk pembentukan label data *testing*, disediakan data variabel *matlab StimulusCode* seperti yang dijelaskan pada Tabel 3.1 dan 3.2, kode nomor untuk setiap baris/kolom pada matriks seperti pada Gambar 3.1, dan data urutan karakter target. Langkah pertama pada pembentukan label data *testing* yaitu melakukan pengambilan kode nomor baris/kolom untuk setiap intensifikasi dengan iterasi pada Kode sumber 4.5 baris ke-4 sampai 11 yang diimplementasikan menggunakan *matlab*.

```
1.  i = 1;
2.  j = 1;
3.  code = [];
4.  while i < 101
5.    while j < 7542
6.      code = cat(1, code, StimulusCode(i,j));
7.      j = j + 42;
8.    end
9.    i = i + 1;
10.   j = 1;
11. end
```

Kode Sumber 4.6 Pengambilan kode nomor baris/kolom untuk setiap intensifikasi

Setelah didapatkan kode nomor baris/kolom untuk setiap data sinyal, dilakukan konversi kode tersebut ke angka 0 dan 1, dimana angka 0 menunjukkan kelas *non-target* dan angka 1 menunjukkan kelas *target*. Proses konversi dilakukan dengan cara melihat karakter *target*, dan kode nomor baris/kolom pada intensifikasi saat ini. Jika dalam baris/kolom kode nomor tersebut terdapat karakter *target*, maka kode nomor tersebut akan disubstitusikan dengan angka 1. Jika sebaliknya akan disubstitusi dengan angka 0, yang dilakukan oleh kode sumber baris ke-22 sampai 25. Proses ini diimplementasikan menggunakan *Python*

dengan memanfaatkan tipe data *dict* [19] untuk melakukan pencocokan kode baris/kolom dengan karakter *target*.

```
1.  import pandas as pd
2.  code = pd.read_csv("path_to_code",
    header=None)
3.  char = pd.read_csv("path_to_target_char",
    header=None)
4.  label = pd.DataFrame()
5.
6.  speller = {
7.          1: ["A", "G", "M", "S", "Y", "5"],
8.          2: ["B", "H", "N", "T", "Z", "6"],
9.          3: ["C", "I", "O", "U", "1", "7"],
10.         4: ["D", "J", "P", "V", "2", "8"],
11.         5: ["E", "K", "Q", "W", "3", "9"],
12.         6: ["F", "L", "R", "X", "4", "_"],
13.         7: ["A", "B", "C", "D", "E", "F"],
14.         8: ["G", "H", "I", "J", "K", "L"],
15.         9: ["M", "N", "O", "P", "Q", "R"],
16.         10: ["S", "T", "U", "V", "W", "X"],
17.         11: ["Y", "Z", "1", "2", "3", "4"],
18.         12: ["5", "6", "7", "8", "9", "_"]}
19. j = 0;
20.
21. for i in range(code.size):
22.     if char.loc[j,0] in
    speller[code.loc[i,0]] :
23.         label.loc[i,0] = 1
24.     else:
25.         label.loc[i,0] = 0
26.
27.     if i%180 == 179:
28.         j+=1
```

```
29.
30. label.to_csv("destination_path",
    index=False, header=False)
```

Kode Sumber 4.7 Konversi kode nomor baris/kolom ke kode kelas

4.2.3. Implementasi Reduksi Dimensi

Dalam proses reduksi dimensi spasial, diperlukan transformasi data sinyal ke data 2 dimensi, dimana baris pada data menyimpan sinyal temporal di seluruh intensifikasi, dan kolom menyimpan data sinyal pada dimensi spasial. Setelah itu, dilakukan normalisasi sinyal menggunakan metode normalisasi *zero mean*, kemudian barulah dilakukan reduksi dimensi spasial menggunakan *Principal Component Analysis*. Setelah dilakukan *Principal Component Analysis*, dilakukan reduksi dimensi temporal menggunakan *Linear Discriminant Analysis*. Oleh karena itu, data sinyal ditransformasi balik ke representasi semula sebelum dilakukan *Linear Discriminant Analysis*.

4.2.3.1. Implementasi Transformasi Sinyal

Proses reduksi dimensi akan diimplementasikan menggunakan *Python*, sehingga data masukan harus dalam format *comma separated value* dua dimensi. Data sinyal hasil *averaging* disimpan dalam format variabel *matlab* tiga dimensi, sehingga perlu dilakukan pengubahan representasi data ke dalam dua dimensi dan format *comma separated value*. Hal ini dapat dilakukan menggunakan fungsi "*csvwrite*" pada *Matlab* [20].

Untuk setiap data sinyal, dilakukan transformasi sehingga setiap baris pada data merepresentasikan *channel*, dan kolom menyimpan sinyal temporal untuk seluruh intensifikasi dengan iterasi pada baris ke-6 sampai dengan 15. Setelah itu dilakukan transpose pada baris ke-17 sehingga kolom kini menyimpan rekaman sinyal untuk setiap *channel*.

```
1.  load('path_to_averaged_signal')
2.  i = 1;
```

```
3.  j = 1;
4.  transformed_flash = [];
5.  transformed_signal = [];
6.  while i < size(averaged_signal, 1)+1
7.    while j < size(averaged_signal, 3)+1
8.      transformed_flash = cat(1,
    transformed_flash, averaged_signal(i,:,j));
9.      j = j+1;
10.   end
11.   transformed_signal = cat(2,
    transformed_signal, transformed_flash);
12.   transformed_flash = [];
13.   j = 1;
14.   i = i+1;
15. end
16.
17. transformed_signal =
    transpose(transformed_signal);
18. csvwrite('destination_path',
    transformed_signal);
```

Kode Sumber 4.8 Transformasi sinyal ke dua dimensi dan *channel* sebagai kolom

4.2.3.2. Implementasi Normalisasi *Zero Mean*

Kode sumber normalisasi *zero mean* menerima masukan data sinyal hasil transformasi yang menjadi keluaran Kode sumber 4.7. Implementasi normalisasi *zero mean* terdapat pada baris ke-6 sampai 8 pada Kode sumber 4.8 dengan memanfaatkan fungsi *StandardScaler* pada *library scikit-learn* di *Python* [21]. Hasil dari normalisasi *zero mean* kemudian disimpan sebagai data dua dimensi dengan format *comma separated value* pada Kode sumber 4.7 baris ke-9 dan 10.

```
1.  import pandas as pd
2.  from sklearn.preprocessing import
    StandardScaler
3.  train_data = pd.read_csv("train_data_path",
    header=None)
4.  test_data = pd.read_csv("test_data_path",
    header=None)
5.  scaler = StandardScaler()
6.  scaler.fit(train_data)
7.  train_data = scaler.transform(train_data)
8.  test_data = scaler.transform(test_data)
9.  train_data.to_csv(filename, index=False,
    header=False)
10. test_data.to_csv(filename, index=False,
    header=False)
```

Kode Sumber 4.9 Normalisasi *zero mean*

4.2.3.3. Implementasi *Principal Component Analysis*

Kode sumber untuk proses *Principal Component Analysis* menerima masukan berupa data sinyal yang sudah melewati proses transformasi pada subbab 4.2.3.1 dan normalisasi *zero mean* pada subbab 4.2.3.2. Proses *Principal Component Analysis* dimulai dari Kode sumber 4.9 baris ke-6 dan 7, yaitu menemukan *eigenvector* menggunakan data *training*, dengan parameter jumlah *channel* dari data yang ingin dipertahankan. Setelah itu, dilakukan transformasi untuk data *training* dan *testing* menggunakan *eigenvector* yang sudah ditemukan dengan Kode sumber 4.7 baris ke-7 dan 10. Implementasi *Principal Component Analysis* memanfaatkan fungsi *PCA* pada *library scikit-learn* di *Python* [22].

```
1.  import pandas as pd
2.  from sklearn.preprocessing import
    StandardScaler
```

```
3.  train_data = pd.read_csv("train_data_path",
    header=None)
4.  test_data = pd.read_csv("test_data_path",
    header=None)
5.  pca = PCA(channel)
6.  pca.fit(train_data)
7.  train_data = pca.transform(train_data)
8.  train_data = pd.DataFrame(data=train_data)
9.  train_data.to_csv(filename, index=False,
    header=False)
10. test_data = pca.transform(test_data)
11. test_data = pd.DataFrame(data=test_data)
12. test_data.to_csv(filename, index=False,
    header=False)
```

Kode Sumber 4.10 ***Principal Component Analysis***

4.2.3.4. Implementasi Transformasi Balik Sinyal

Sebelum melakukan reduksi dimensi temporal menggunakan *Linear Discriminant Analysis*, dilakukan transformasi balik ke representasi data sebelum dilakukan *Principal Component Analysis*, sehingga data temporal sinyal disimpan pada setiap kolom pada data. Proses transformasi balik dilakukan dengan mentranspose data terlebih dahulu dengan Kode sumber 4.10 baris ke-6, dilanjutkan dengan iterasi pada Kode sumber 4.10 baris ke-7 sampai 16. Karena melibatkan operasi transpose matriks yang membutuhkan fungsi *transpose*, transformasi balik diimplementasikan menggunakan *Matlab*.

```
1.  transformed_signal =
    csvread('path_to_data');
2.  reduced_signal = [];
3.  reduced_flash = [];
4.  i = 1;
5.  j = 1;
```

```
6.  transformed_signal =
    transpose(transformed_signal);
7.  while i < size(transformed_signal, 1)+1
8.    while j < size(transformed_signal, 2)+1
9.      reduced_flash = cat(1, reduced_flash,
    transformed_signal(i, j:j+14));
10.         j = j+15;
11.   end
12.   reduced_signal = cat(3, reduced_signal,
    reduced_flash);
13.   reduced_flash = [];
14.   j = 1;
15.   i = i+1;
16. end
17. csvwrite('destination_path',
    reduced_signal);
```

Kode Sumber 4.11 Transformasi balik sinyal

4.2.3.5. Implementasi *Linear Discriminant Analysis*

Kode sumber proses *Linear Discriminant Analysis* menerima masukan berupa data sinyal dua dimensi, dimana kolom merepresentasikan sinyal temporal untuk seluruh *channel*, dan baris merepresentasikan masing-masing intensifikasi. Proses pencarian *eigenvector* pada proses ini menggunakan data *training* beserta labelnya, seperti pada Kode sumber 4.11 baris ke-6 dan 7. Kemudian transformasi dilakukan untuk data *training* dan *testing* dengan kode sumber baris ke-8 dan 9. Implementasi proses ini menggunakan fungsi *LDA* pada *library scikit-learn* di *Python* [23].

```
1.  import pandas as pd
2.  from sklearn.discriminant_analysis import
    LinearDiscriminantAnalysis as LDA
3.  test = pd.read_csv("test_data_path",
    header=None)
```

```
4.  train = pd.read_csv("train_data_path",
    header=None)
5.  label = pd.read_csv("label_path",
    header=None)
6.  lda = LDA()
7.  lda.fit(train, label)
8.  train = pd.DataFrame(lda.transform(train))
9.  test = pd.DataFrame(lda.transform(test))
10. train.to_csv(filename, index=False,
    header=False)
11. test.to_csv(filename, index=False,
    header=False)
```

Kode Sumber 4.12 *Linear Discriminant Analysis*

4.2.4. Implementasi *Balancing* Antar Kelas

Pada proses *balancing* antar kelas pada data sinyal, dilakukan eksperimen menggunakan tiga algoritma berbeda, yaitu *Borderline SMOTE*, *ADASYN*, dan duplikasi. Implementasi ketiga algoritma tersebut menggunakan *Python* yang akan dibahas dalam subbab 4.2.4.1 hingga 4.2.4.3.

4.2.4.1. Implementasi *Balancing* dengan *Borderline SMOTE*

Proses *balancing* dengan *Borderline SMOTE* diimplementasikan dengan fungsi *SMOTE* pada *library imbalanced-learn* [24]. Kode sumber *balancing* dengan *Borderline SMOTE* menerima data, label dari data tersebut dan banyaknya data per kelas sebagai masukan. Banyaknya data per kelas didefinisikan pada kode sumber baris ke-3. Proses *balancing* dilakukan oleh kode sumber baris ke-6 yang menghasilkan data yang jumlah anggota per kelasnya sesuai dengan yang sudah didefinisikan pada kode baris ke-3.

```
1.  import pandas as pd
2.  from imblearn.over_sampling import SMOTE
```

```
3.  smote = SMOTE (ratio={1:2550},
    kind='borderline1')
4.  data = pd.read_csv("data_path",
    header=None)
5.  label = pd.read_csv("label_path",
    header=None)
6.  data_resampled, label_resampled =
    smote.fit_sample(data, label)
7.  data_resampled =
    pd.DataFrame(data_resampled)
8.  label_resampled =
    pd.DataFrame(label_resampled)
9.  data_resampled.to_csv("resampled_data_path"
    , header=False, index=False)
10. label_resampled.to_csv("resampled_label_pat
    h", header=False, index=False)
```

Kode Sumber 4.13 *Balancing* dengan *Borderline SMOTE*

4.2.4.2. Implementasi *Balancing* dengan *ADASYN*

Proses *balancing* dengan algoritma *ADASYN* diimplementasikan menggunakan fungsi *ADASYN* pada *library imbalanced-learn* [25]. Kode sumber untuk *balancing* dengan *ADASYN* menerima masukan berupa data, label dari data tersebut, dan parameter jumlah anggota untuk setiap kelasnya. Parameter jumlah anggota per kelas didefinisikan pada kode sumber baris ke-3. Proses *balancing* dilakukan oleh kode sumber baris ke-6, yang menghasilkan data dan label dengan jumlah anggota per kelas sesuai dengan parameter yang sudah didefinisikan pada kode sumber baris ke-3.

```
1.  import pandas as pd
2.  from imblearn.over_sampling import ADASYN
3.  smote = ADASYN(ratio={1:2550})
4.  data = pd.read_csv("data_path",
    header=None)
```

```
5.  label = pd.read_csv("label_path",
    header=None)
6.  data_resampled, label_resampled =
    smote.fit_sample(data, label)
7.  data_resampled =
    pd.DataFrame(data_resampled)
8.  label_resampled =
    pd.DataFrame(label_resampled)
9.  data_resampled.to_csv("resampled_data_path"
    , header=False, index=False)
10. label_resampled.to_csv("resampled_label_pat
    h", header=False, index=False)
```

Kode Sumber 4.14 *Balancing* dengan *ADASYN*

4.2.4.3. Implementasi *Balancing* dengan Duplikasi

Kode sumber proses *balancing* dengan duplikasi menerima masukan berupa data sinyal beserta labelnya. Proses duplikasi dilakukan dengan mengumpulkan data yang ada di kelas minor (*target*) dengan kode sumber baris ke-6, kemudian dilakukan duplikasi data pada kelas minor (*target*) dengan metode *round robin* yang diimplementasikan dengan kode sumber baris ke-8 sampai 15, sehingga antar masing-masing data asli pada kelas minor (*target*) memiliki maksimal selisih duplikat sebanyak satu data.

```
1.  data = pd.read_csv("data_path",
    header=None)
2.  label = pd.read_csv("label_path",
    header=None)
3.  target_data = (label==1).sum()[0]
4.  non_target_data = (label==0).sum()[0]
5.  duplicate_count = non_target_data -
    target_data
6.  index_target_data = label.index[label[0] ==
    1].tolist()
7.  i = 0
```

```
8.  while duplicate_count > 0 :
9.      data =
    data.append(data.loc[index_target_data[i]],
    ignore_index=True)
10.     label =
    label.append(label.loc[index_target_data[i]
    ], ignore_index=True)
11.     duplicate_count -= 1
12.     if i >= len(index_target_data) - 1 :
13.         i = 0
14.     else :
15.         i+=1
16. data_resampled.to_csv("resampled_data_path"
    , header=False, index=False)
17. label_resampled.to_csv("resampled_label_pat
    h", header=False, index=False)
```

Kode Sumber 4.15 *Balancing* dengan duplikasi

4.2.5. Implementasi Pembuatan dan Evaluasi Kinerja *Model* Klasifikasi *Support Vector Machine*

Kode sumber proses pembuatan *model* klasifikasi dan evaluasi kinerja menerima masukan berupa data sinyal *training* dan *testing* yang setidaknya sudah melewati seluruh proses *preprocessing* sinyal pada subbab 4.2.1, beserta label dari masing-masing data *training* dan *testing*. Proses pembuatan model dan prediksi dilakukan menggunakan kelas *SVC* pada *library scikit-learn* di *Python* pada kode sumber baris ke-8 sampai 9 [26]. Setelah itu, dilanjutkan dengan evaluasi dari *model* yang sudah dibuat menggunakan kelas *classification_report* dan *confusion_matrix* dari library *scikit-learn*. Proses evaluasi dimulai dengan membuat prediksi kelas dari data *testing* dengan kode sumber pada baris ke-10, kemudian dilakukan pembuatan *confusion matrix* dengan kode sumber baris ke-11 dan penghitungan *precision*, *recall*, dan *f1-score* masing-masing kelas menggunakan kode sumber baris ke-12.

```
1.  import pandas as pd
2.  from sklearn.svm import SVC
3.  from sklearn.metrics import
    classification_report,confusion_matrix
4.  train_data = pd.read_csv("train_data_path",
    header=None)
5.  train_label =
    pd.read_csv("train_label_path",
    header=None)
6.  test_data = pd.read_csv("test_data_path",
    header=None)
7.  test_label = pd.read_csv("test_label_path",
    header=None)
8.  svm = SVC(kernel='linear')
9.  svm.fit(train_data, train_label)
10. prediction = svm.predict(test_data)
11. print(confusion_matrix(test_label,
    prediction))
12. print(classification_report(test_label,
    prediction))
```

Kode Sumber 4.16 Pembuatan dan evaluasi *model* klasifikasi *Support Vector Machine*

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

5.1. Lingkungan Pengujian

Lingkungan uji coba yang akan digunakan untuk klasifikasi sinyal P300 mencakup perangkat keras dan perangkat lunak. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Uji Coba Perangkat Keras dan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel ® Core i7-8550U CPU @ 1.80GHz (8 CPUs), ~2.0GHz Memori: 16384MB RAM
Perangkat lunak	Sistem Operasi: • Windows 10 Pro 64-bit Perangkat Pengembang: • MATLAB 7.8.0.347 (R2009a) • Python 3.6.4 • pandas 0.22.0 • scikit-learn 0.19.1 • imbalanced-learn 0.3.1 Perangkat Pembantu: • Spyder 3.3.1

5.2. Data Uji Coba

Data yang digunakan dalam proses uji coba klasifikasi sinyal P300 adalah Wadsworth BCI Dataset (P300 Evoked Potentials), BCI Competition III Challenge 2004. Dataset ini berisi rekaman sinyal EEG dari dua subjek berbeda, untuk masing-masing subjek sudah disediakan data *training* berupa rekaman sinyal EEG untuk

85 karakter target, dan data *testing* untuk 100 karakter target. Seluruh data *training* dan *testing* akan melewati proses *preprocessing* sinyal yang sudah diimplementasikan pada subbab 4.2.1. Setelah itu, dilakukan reduksi dimensi dan pembuatan *model* dengan berbagai skenario uji coba yang akan dibahas pada subbab 5.4.

5.3. Hasil Uji Coba *Preprocessing* Sinyal dan Pembentukan Label Data

5.3.1. Eliminasi *Channel*, Pemotongan Sinyal EEG, dan *Butterworth Bandpass Filter*

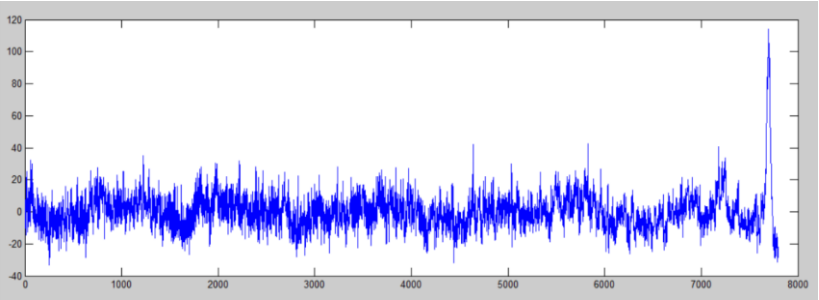
Proses uji coba eliminasi *channel*, pemotongan sinyal EEG dan *Butterworth Bandpass Filter* menerima masukan berupa data dengan format variabel *Matlab* bernama *signal* dengan ukuran untuk data *training* dan *testing* kedua subjek yang disajikan dalam Tabel 5.2. Waktu eksekusi yang dibutuhkan untuk memproses setiap data dapat dilihat dalam Tabel 5.3. Sinyal yang menjadi masukan dalam proses ini adalah kumpulan sinyal kontinyu seperti pada Gambar 5.1. Kemudian hasil pemotongan sinyal EEG untuk setiap intensifikasi memiliki panjang sebanyak 160 sampel sinyal seperti pada Gambar 5.2, yang didapat dari dari *channel* FC_5 selama 667 ms sejak dimulainya intensifikasi pertama dari data *training* subjek A. *Butterworth Bandpass Filter* kemudian diaplikasikan pada sinyal hasil dari pemotongan, sehingga menghasilkan sinyal seperti pada Gambar 5.2, yang merupakan hasil filter dari sinyal yang ada pada Gambar 5.1. Terlihat bahwa *random noise* pada sinyal sudah berkurang. Keluaran dari proses ini adalah variabel *Matlab* dengan ukuran yang disajikan pada Tabel 5.2.

Tabel 5.2 Ukuran Data Masukan dan Keluaran Masing-Masing Data yang Melewati Proses Eliminasi *Channel*, Pemotongan Sinyal, dan *Butterworth Bandpass Filter*

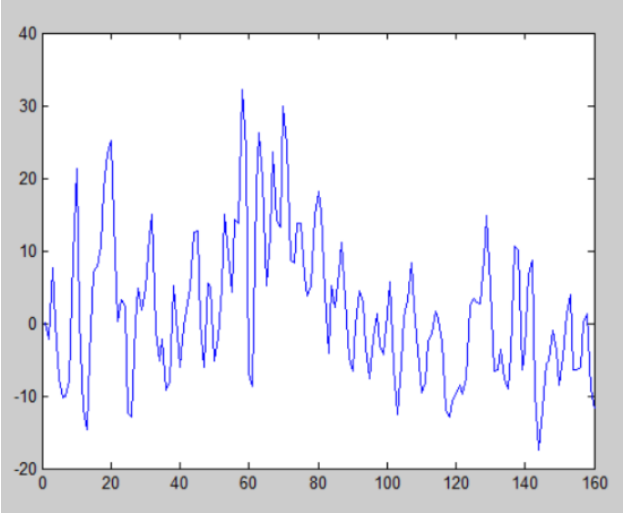
No	Data	Ukuran	
		Data masukan	Data keluaran
1.	*Training* subjek A	85x7794x64	15300x160x51
2.	*Testing* subjek A	100x7794x64	18000x160x51
3.	*Training* subjek B	85x7794x64	15300x160x51
4.	*Testing* subjek B	100x7794x64	18000x160x51

Tabel 5.3 Waktu Eksekusi untuk Proses Eliminasi *Channel*, Pemotongan Sinyal EEG, dan *Butterworth Bandpass Filter* untuk Setiap Data

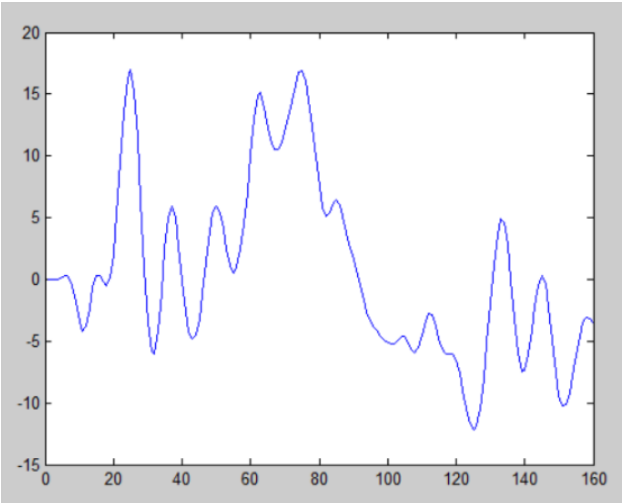
No	Data	Waktu eksekusi (dalam detik)
1.	*Training* subjek A	4.376
2.	*Testing* subjek A	6.099
3.	*Training* subjek B	4.663
4.	*Testing* subjek B	6.071



Gambar 5.1 Contoh data masukan satu sinyal kontinyu, dari *channel* FC_5 intensifikasi karakter pertama pada data *training* subjek A



Gambar 5.2 Contoh rekaman sinyal EEG sebelum aplikasi *Butterworth Bandpass Filter*



Gambar 5.3 Contoh rekaman sinyal EEG setelah aplikasi *Butterworth Bandpass Filter*

5.3.2. *Signal Downsampling*

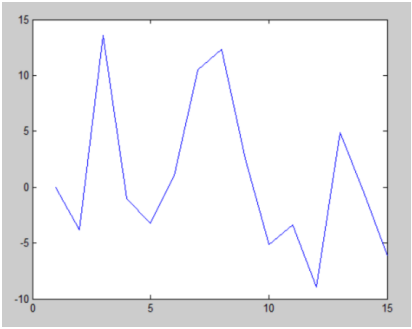
Proses uji coba proses *signal downsampling* menerima data masukan berupa data *training* dan *testing* kedua subjek dari hasil uji coba 5.3.1 dengan ukuran tiap data yang disajikan pada Tabel 5.4. Waktu eksekusi dari proses ini dapat dilihat pada Tabel 5.5. Pada Gambar 5.4, ditampilkan hasil *signal downsampling* yang diaplikasikan pada sinyal yang ada di Gambar 5.3.

Tabel 5.4 Ukuran Data Masukan dan Keluaran Masing-Masing Data yang Melalui Proses *Signal Downsampling*

No	Data	Ukuran	
		Data masukan	Data keluaran
1.	*Training* subjek A	15300x160x51	15300x15x51
2.	*Testing* subjek A	18000x160x51	18000x15x51
3.	*Training* subjek B	15300x160x51	15300x15x51
4.	*Testing* subjek B	18000x160x51	18000x15x51

Tabel 5.5 Waktu Eksekusi Proses *Signal Downsampling*

No	Data	Waktu eksekusi (dalam detik)
1.	*Training* subjek A	424
2.	*Testing* subjek A	571
3.	*Training* subjek B	458
4.	*Testing* subjek B	607



Gambar 5.4 Contoh rekaman sinyal EEG setelah proses *signal downsampling*

5.3.3. *Signal Averaging*

Proses uji coba proses *signal averaging* dilakukan dengan memasukkan data hasil proses *signal downsampling* pada uji coba 5.3.2. Spesifikasi ukuran masing-masing data masukan dan keluaran dari proses *signal averaging* dapat dilihat pada Tabel 5.5. Waktu eksekusi untuk setiap data dapat dilihat pada Tabel 5.8. Pada Gambar 5.5 ditampilkan lima potongan sinyal EEG pertama dari kelas *non-target* pada data *training* subjek A. Kemudian dilakukan proses *signal averaging* yang menghasilkan keluaran satu buah potongan sinyal EEG seperti pada Gambar 5.6. Selain *averaging* data sinyal, perlu dilakukan pemberian label untuk setiap data potongan sinyal hasil *averaging*, sehingga proses signal averaging juga menerima masukan label dari masing-masing data dan menghasilkan keluaran dengan spesifikasi ukuran seperti pada Tabel 5.7.

Tabel 5.6 Ukuran Data Masukan dan Keluaran Masing-Masing Data yang Melalui Proses *Signal Averaging*

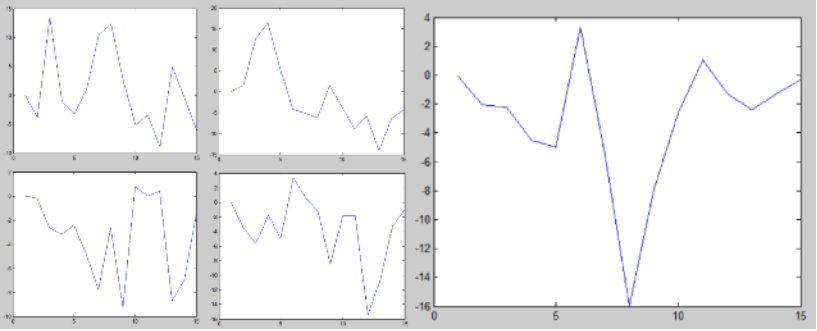
No	Data	Ukuran	
		Data masukan	Data keluaran
1.	*Training* subjek A	15300x15x51	3060x15x51
2.	*Testing* subjek A	18000x15x51	3600x15x51
3.	*Training* subjek B	15300x15x51	3060x15x51
4.	*Testing* subjek B	18000x15x51	3600x15x51

Tabel 5.7 Ukuran Label Data Masukan dan Keluaran Masing-Masing Data yang Melalui Proses Signal Averaging

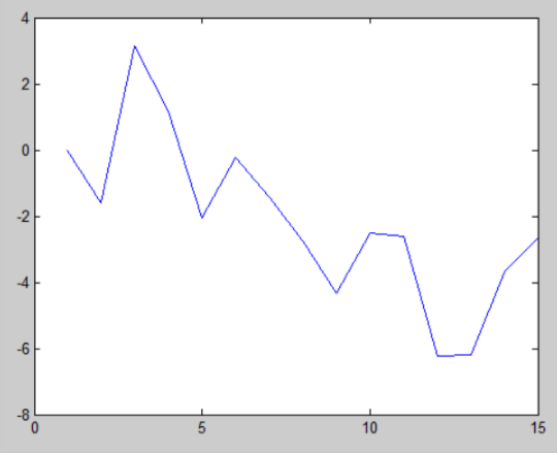
No	Data	Ukuran	
		Data masukan	Data keluaran
1.	*Training* subjek A	15300x1	3060x1
2.	*Testing* subjek A	18000x1	3600x1
3.	*Training* subjek B	15300x1	3060x1
4.	*Testing* subjek B	18000x1	3600x1

Tabel 5.8 Waktu Eksekusi Proses *Signal Averaging*

No	Data	Waktu eksekusi (dalam detik)
1.	*Training* subjek A	12.29
2.	*Testing* subjek A	17.19
3.	*Training* subjek B	12.92
4.	*Testing* subjek B	17.10



Gambar 5.5 Contoh masukan sinyal EEG yang akan melalui proses *signal averaging*



Gambar 5.6 Contoh sinyal EEG hasil dari proses *signal averaging* sinyal-sinyal yang ada di Gambar 5.5

5.3.4. Pembentukan Label Data *Training*

Proses uji coba pembentukan label data *training* dilakukan dengan memasukkan data berupa variabel *Matlab* dua dimensi bernama *StimulusType* dari kedua subjek. Spesifikasi ukuran data masukan dan keluaran dapat dilihat pada Tabel 5.9. Waktu eksekusi untuk proses pembentukan label data *training* untuk masing-masing subjek dapat dilihat pada Tabel 5.10.

Tabel 5.9 Ukuran Data Masukan dan Keluaran dari Proses Pembentukan Label Data *Training*

No	Data	Ukuran	
		Data masukan	Data keluaran
1.	*Training* subjek A	85x7794	15300x1
2.	*Training* subjek B	85x7794	15300x1

Tabel 5.10 Waktu Eksekusi Proses Pembentukan Label Data *Training*

No	Data	Waktu eksekusi (dalam detik)
1.	*Training* subjek A	0.16
3.	*Training* subjek B	0.13

5.3.5. Pembentukan Label Data *Testing*

Proses uji coba pembentukan label data *testing* dilakukan dengan memasukkan variabel matlab bernama *StimulusCode* dan daftar karakter target untuk masing-masing subjek. Spesifikasi ukuran data masukan dan keluaran dapat dilihat pada Tabel 5.11. Waktu eksekusi untuk proses ini dapat dilihat pada Tabel 5.12.

Tabel 5.11 Ukuran Data Masukan dan Keluaran dari Proses Pembentukan Label Data *Testing*

No	Data	Ukuran		
		Data masukan	Daftar karakter target	Data keluaran
1.	*Testing* subjek A	100x7794	100x1	15300x1
2.	*Testing* subjek B	100x7794	100x1	15300x1

Tabel 5.12 Waktu Eksekusi Proses Pembentukan Label Data *Testing*

No	Data	Waktu eksekusi (dalam detik)	
		Pengambilan kode baris/kolom	Konversi ke kode kelas
1.	*Testing* subjek A	0.17	12.63
3.	*Testing* subjek B	0.19	12.59

5.4. Skenario Uji Coba

Pada subbab ini akan dijelaskan skenario-skenario uji coba yang akan dilakukan dalam proses klasifikasi sinyal P300 dari data *training* dan data *testing* masing-masing subjek yang sudah melalui proses *preprocessing* sinyal. Skenario-skenario uji coba meliputi variasi jumlah *channel* yang akan dipertahankan dalam proses reduksi dimensi spasial (*channel*) menggunakan metode *Principal Component Analysis*, penerapan *Linear Discriminant Analysis* dalam reduksi dimensi temporal, dan variasi algoritma *balancing* antar kelas yang dilakukan. Skenario-skenario yang dilakukan dalam proses uji coba yaitu :

1. Klasifikasi tanpa dilakukan *balancing*, tanpa reduksi dimensi *PCA* dan *LDA*.

2. Klasifikasi tanpa dilakukan *balancing*, dengan reduksi dimensi *PCA* saja.
3. Klasifikasi tanpa dilakukan *balancing*, dengan reduksi dimensi *PCA* dan *LDA*.
4. Klasifikasi setelah *balancing*, tanpa reduksi dimensi.
5. Klasifikasi setelah *balancing*, dengan reduksi dimensi *PCA* saja.
6. Klasifikasi setelah *balancing*, dengan reduksi dimensi *PCA* dan *LDA*.

Untuk seluruh skenario uji coba akan dilakukan evaluasi sesuai dengan desain yang sudah dibuat pada subbab 3.2.6, dimana kelas 0 merupakan kelas *non-target*, dan kelas 1 merupakan kelas *target*.

5.4.1 Skenario Uji Coba 1

Pada skenario uji coba 1, dilakukan proses pembuatan *model* dan klasifikasi dengan menggunakan data *training* dan *testing* tanpa dilakukan *balancing* dan reduksi dimensi apapun. Untuk data *training* setiap subjek, terdapat 2.550 data dari kelas *non-target* (0) dan 510 data dari kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari kelas *non-target* (0) dan 600 data dari kelas *target* (1). Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel*. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini adalah 3,9 detik untuk subjek A dan 3,07 detik Untuk subjek B. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.13 dan 5.14. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran. Pada skenario uji coba 1, dilakukan proses klasifikasi dengan menggunakan data *training* dan *testing* tanpa dilakukan *balancing* dan reduksi dimensi apapun.

Tabel 5.13 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 1 pada Subjek A

Kelas	*Precision*	*Recall*	*F1-score*
0	94%	91%	93%
1	62%	71%	66%
Rata-rata	89%	88%	88%

Tabel 5.14 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 1 pada Subjek B

Kelas	*Precision*	*Recall*	*F1-score*
0	97%	90%	93%
1	63%	85%	73%
Rata-rata	91%	89%	90%

5.4.2. Skenario Uji Coba 2

Pada skenario uji coba 2, dilakukan pembuatan *model* dan klasifikasi dengan data *training* dan *testing* tanpa dilakukan *balancing*, dengan reduksi dimensi menggunakan metode *PCA*. Untuk data *training* setiap subjek, terdapat 2.550 data dari kelas *non-target* (0) dan 510 data dari kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari kelas *non-target* (0) dan 600 data dari kelas *target* (1). Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel* yang akan direduksi dengan *PCA*. Terdapat beberapa variasi jumlah *channel* data sinyal EEG yang dipertahankan pada skenario ini, yaitu :

1. Jumlah *channel* sebanyak 46
2. Jumlah *channel* sebanyak 40
3. Jumlah *channel* sebanyak 33
4. Jumlah *channel* sebanyak 26
5. Jumlah *channel* sebanyak 19
6. Jumlah *channel* sebanyak 13
7. Jumlah *channel* sebanyak 8

Proses reduksi dimensi dihentikan ketika jumlah *channel* berjumlah sebanyak 8. Keputusan ini diambil dikarenakan tidak adanya peningkatan performa klasifikasi seiring dilakukannya reduksi dimensi *channel*. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.15 dan 5.16. Waktu yang dibutuhkan untuk proses *PCA* dapat dilihat pada Tabel 5.17, dan proses *training* pada Tabel 5.18. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.15 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 2 pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	91%	92%
1	61%	72%	66%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	62%	70%	66%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	63%	71%	66%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	64%	70%	67%
Rata-rata	89%	88%	89%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*

0	94%	92%	93%
1	62%	69%	66%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	65%	70%	67%
Rata-rata	89%	89%	89%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	93%	92%	93%
1	63%	66%	65%
Rata-rata	88%	88%	88%

Tabel 5.16 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 2 pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	90%	93%
1	63%	85%	72%
Rata-rata	91%	89%	90%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	89%	93%
1	62%	84%	71%
Rata-rata	91%	89%	89%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	96%	89%	93%
1	60%	84%	70%
Rata-rata	90%	88%	89%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*

0	96%	90%	93%
1	61%	81%	70%
Rata-rata	90%	88%	89%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	96%	88%	92%
1	59%	82%	69%
Rata-rata	90%	87%	88%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	95%	88%	92%
1	57%	78%	66%
Rata-rata	89%	87%	87%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	88%	91%
1	55%	71%	62%
Rata-rata	87%	85%	86%

Tabel 5.17 Waktu Eksekusi Proses *PCA* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 2

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.20	0.23
2.	40	0.70	0.78
3.	33	0.70	0.75
4.	26	0.54	0.64
5.	19	0.51	0.53
6.	13	0.39	0.59
7.	8	0.37	0.39

Tabel 5.18 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 2

No	Jumlah *channel*	Waktu eksekusi (dalam detik)

		Subjek A	Subjek B
1.	46	4.10	2.92
2.	40	4.34	2.65
3.	33	4.68	2.82
4.	26	5.81	4.58
5.	19	9.71	5.37
6.	13	7.48	8.24
7.	8	6.42	8.23

5.4.3. Skenario Uji Coba 3

Pada skenario uji coba 3, dilakukan pembuatan *model* dan klasifikasi dengan data *training* dan *testing* tanpa dilakukan *balancing*, dengan reduksi dimensi menggunakan metode *PCA* dan *LDA*. Untuk data *training* setiap subjek, terdapat 2.550 data dari kelas *non-target* (0) dan 510 data dari kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari kelas *non-target* (0) dan 600 data dari kelas *target* (1). Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel* yang akan direduksi dengan *PCA*. Terdapat beberapa variasi jumlah *channel* data sinyal EEG yang dipertahankan pada proses *PCA*, yaitu :

1. Jumlah *channel* sebanyak 46
2. Jumlah *channel* sebanyak 40
3. Jumlah *channel* sebanyak 33
4. Jumlah *channel* sebanyak 26
5. Jumlah *channel* sebanyak 19
6. Jumlah *channel* sebanyak 13
7. Jumlah *channel* sebanyak 8

Kemudian pada masing-masing data tersebut diaplikasikan metode reduksi dimensi *LDA*. Proses *LDA* mereduksi dimensi sehingga data keluarannya hanya memiliki satu fitur. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.19 dan 5.20. Waktu yang dibutuhkan untuk proses *LDA* dapat dilihat pada Tabel 5.21, dan proses *training* pada Tabel 5.22. Pada beberapa kasus, proses *training* selesai secara instan sehingga waktu eksekusinya tidak

terekam. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.19 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 3 pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	91%	93%
1	61%	73%	67%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	95%	92%	93%
1	64%	74%	69%
Rata-rata	90%	89%	89%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	65%	71%	68%
Rata-rata	89%	89%	89%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	95%	92%	93%
1	64%	74%	69%
Rata-rata	90%	89%	89%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	92%	93%
1	63%	72%	67%
Rata-rata	89%	88%	89%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	91%	93%

1	62%	72%	67%
Rata-rata	89%	88%	88%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	93%	93%	93%
1	64%	65%	64%
Rata-rata	88%	88%	88%

Tabel 5.20 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 3 pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	90%	93%
1	63%	88%	73%
Rata-rata	92%	89%	90%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	90%	93%
1	63%	88%	73%
Rata-rata	92%	89%	90%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	90%	93%
1	63%	85%	72%
Rata-rata	91%	89%	90%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	91%	94%
1	64%	85%	73%
Rata-rata	91%	90%	90%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	96%	89%	92%

1	60%	82%	69%
Rata-rata	90%	88%	88%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	96%	89%	92%
1	59%	81%	68%
Rata-rata	90%	87%	88%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	88%	91%
1	56%	73%	63%
Rata-rata	88%	86%	86%

Tabel 5.21 Waktu Eksekusi Proses *LDA* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 3

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.37	0.34
2.	40	0.30	0.26
3.	33	0.23	0.23
4.	26	0.14	0.12
5.	19	0.09	0.09
6.	13	0.06	0.05
7.	8	0.04	0.03

Tabel 5.22 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 3

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.0	0.0
2.	40	0.004	0.0
3.	33	0.0	0.0
4.	26	0.0	0.0
5.	19	0.01	0.01

6.	13	0.01	0.0
7.	8	0.01	0.01

5.4.4. Skenario Uji Coba 4

Pada skenario uji coba 4, dilakukan proses pembuatan *model* dan klasifikasi dengan data *training* dan *testing* yang sudah melewati proses *balancing* antar kelas tanpa reduksi dimensi. Berbagai metode *balancing* antar kelas akan dilakukan pada subbab 5.4.4.1 sampai 5.4.4.3.

5.4.4.1. Dengan Metode *Balancing Borderline SMOTE*

Proses *balancing* dengan *Borderline SMOTE* diaplikasikan pada data *training* dan *testing* yang belum melewati proses reduksi dimensi memakan waktu eksekusi selama 16,15 detik untuk subjek A, dan 18,60 detik untuk subjek B. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel*. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini adalah 9,23 detik untuk subjek A dan 6,64 detik Untuk subjek B. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.23 dan 5.24. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.23 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan *Borderline SMOTE* pada Subjek A

Kelas	*Precision*	*Recall*	*F1-score*
0	80%	91%	85%
1	90%	77%	83%
Rata-rata	85%	84%	84%

Tabel 5.24 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan *Borderline SMOTE* pada Subjek B

Kelas	*Precision*	*Recall*	*F1-score*
0	91%	90%	90%
1	90%	91%	90%
Rata-rata	90%	90%	90%

5.4.4.2. Dengan Metode *Balancing ADASYN*

Proses *balancing* dengan *ADASYN* diaplikasikan pada data *training* dan *testing* yang belum melewati proses reduksi dimensi memakan waktu eksekusi selama 17,66 detik untuk subjek A, dan 18,94 detik untuk subjek B. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dan *testing* dengan anggota per kelasnya seperti pada Tabel 5.25. Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel*. Jumlah anggota per kelas dapat berbeda-beda pada setiap data karena dilakukan perhitungan jumlah data sintetis yang akan dibuat menggunakan rasio densitas. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini adalah 2.936 detik untuk subjek A dan 1.122 detik Untuk subjek B. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.26 dan 5.27. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.25 Jumlah Data Per Kelas untuk Skenario Uji Coba 4 dengan *ADASYN*

No	Data	Jumlah data per kelas	
		Non-target	*Target*
1.	*Training* subjek A	2.550	2.558
2.	*Testing* subjek A	3.000	2.965
3.	*Training* subjek B	2.550	2.614
4.	*Testing* subjek B	3.000	2.913

Tabel 5.26 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan *ADASYN* pada Subjek A

Kelas	*Precision*	*Recall*	*F1-score*
0	66%	75%	70%
1	70%	60%	65%
Rata-rata	68%	68%	67%

Tabel 5.27 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan *ADASYN* pada Subjek B

Kelas	*Precision*	*Recall*	*F1-score*
0	68%	74%	71%
1	70%	63%	67%
Rata-rata	69%	69%	69%

5.4.4.3. Dengan Metode *Balancing* Duplikasi

Proses *balancing* dengan duplikasi diaplikasikan pada data *training* dan *testing* yang belum melewati proses reduksi dimensi memakan waktu eksekusi selama 139,21 detik untuk subjek A, dan 140,24 detik untuk subjek B. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Seluruh data *training* dan *testing* dari kedua subjek memiliki 51 *channel*. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini adalah 10,85 detik untuk subjek A dan 8,63 detik Untuk subjek B. Evaluasi performa model untuk mengklasifikasi sinyal dari kedua subjek dapat dilihat pada Tabel 5.28 dan 5.29. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.28 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan Duplikasi pada Subjek A

Kelas	*Precision*	*Recall*	*F1-score*
0	76%	91%	83%

1	89%	71%	79%
Rata-rata	83%	81%	81%

Tabel 5.29 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 4 dengan Duplikasi pada Subjek B

Kelas	*Precision*	*Recall*	*F1-score*
0	86%	90%	88%
1	90%	85%	87%
Rata-rata	88%	88%	88%

5.4.5. Skenario Uji Coba 5

Pada skenario uji coba 5, dilakukan proses *balancing* setiap data *training* dan *testing* hasil reduksi dimensi *PCA*. Kemudian dilanjutkan dengan pembuatan *model*, dan klasifikasi. Berbagai metode *balancing* antar kelas akan dilakukan pada subbab 5.4.5.1 sampai 5.4.5.3.

5.4.5.1. Dengan Metode *Balancing Borderline SMOTE*

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 2, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.30. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan *Borderline SMOTE* dari kedua subjek dapat dilihat pada Tabel 5.31 dan 5.32. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.33. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.30 Waktu Eksekusi Proses *Balancing* dengan *Borderline SMOTE* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	4.71	5.87
2.	40	4.59	5.34
3.	33	3.95	4.36
4.	26	3.23	4.09
5.	19	2.52	3.34
6.	13	1.80	2.53
7.	8	1.06	1.64

Tabel 5.31 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan *Borderline SMOTE* pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	80%	91%	85%
1	89%	77%	83%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	79%	91%	85%
1	90%	76%	82%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	80%	90%	85%
1	89%	78%	83%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	81%	88%	84%

1	87%	80%	83%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	86%	85%
1	86%	83%	85%
Rata-rata	85%	85%	85%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	84%	86%
1	85%	87%	86%
Rata-rata	86%	86%	86%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	84%	84%
1	84%	84%	84%
Rata-rata	84%	84%	84%

Tabel 5.32 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan *Borderline SMOTE* pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	90%	90%	90%
1	90%	90%	90%
Rata-rata	90%	90%	90%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	91%	89%	90%
1	90%	91%	90%
Rata-rata	90%	90%	90%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	89%	89%	89%

1	89%	89%	89%
Rata-rata	89%	89%	89%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	88%	87%
1	87%	87%	87%
Rata-rata	87%	87%	87%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	85%	86%
1	85%	88%	86%
Rata-rata	86%	86%	86%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	82%	84%
1	83%	87%	85%
Rata-rata	85%	85%	85%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	79%	82%
1	80%	85%	83%
Rata-rata	82%	82%	82%

Tabel 5.33 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5 dengan *Borderline SMOTE*

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	10.78	6.59
2.	40	10.06	5.32
3.	33	12.07	5.21
4.	26	13.09	7.28
5.	19	15.24	9.67

6.	13	15.77	9.87
7.	8	16.87	12.10

5.4.5.2. Dengan Metode *Balancing ADASYN*

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 2, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.34. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek dapat dilihat pada Tabel 5.35. Jumlah anggota per kelas dapat berbeda-beda pada setiap data karena dilakukan perhitungan jumlah data sintetis yang akan dibuat menggunakan rasio densitas. Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan *ADASYN* dari kedua subjek dapat dilihat pada Tabel 5.36 dan 5.37. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.38. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.34 Waktu Eksekusi Proses *Balancing* dengan *ADASYN* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	5.24	5.89
2.	40	4.85	5.59
3.	33	4.11	4.73
4.	26	3.27	3.92
5.	19	2.70	3.84
6.	13	1.79	2.49
7.	8	1.15	1.74

Tabel 5.35 Jumlah Data Per Kelas untuk Skenario Uji Coba 5 dengan *ADASYN*

No	Data	Jumlah *channel*	Jumlah data per kelas	
			Non-target	*Target*
1.	*Training* subjek A	46	2.550	2.590
2.	*Testing* subjek A		3.000	2.948
3.	*Training* subjek B		2.550	2.602
4.	*Testing* subjek B		3.000	2.897
5.	*Training* subjek A	40	2.550	2.508
6.	*Testing* subjek A		3.000	2.935
7.	*Training* subjek B		2.550	2.527
8.	*Testing* subjek B		3.000	2.893
9.	*Training* subjek A	33	2.550	2.606
10.	*Testing* subjek A		3.000	2.928
11.	*Training* subjek B		2.550	2.537
12.	*Testing* subjek B		3.000	2.890
13.	*Training* subjek A	26	2.550	2.485
14.	*Testing* subjek A		3.000	2.935
15.	*Training* subjek B		2.550	2.626

16.	*Testing* subjek B		3.000	2.890
17.	*Training* subjek A	19	2.550	2.538
18.	*Testing* subjek A		3.000	2.924
19.	*Training* subjek B		2.550	2.558
20.	*Testing* subjek B		3.000	2.903
21.	*Training* subjek A	13	2.550	2.532
22.	*Testing* subjek A		3.000	2.916
23.	*Training* subjek B		2.550	2.539
24.	*Testing* subjek B		3.000	2.926
25.	*Training* subjek A	8	2.550	2.506
26.	*Testing* subjek A		3.000	2.913
27.	*Training* subjek B		2.550	2.548
28.	*Testing* subjek B		3.000	2.950

Tabel 5.36 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan *ADASYN* pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	66%	78%	72%
1	72%	60%	66%
Rata-rata	69%	69%	69%

Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	67%	77%	71%
1	72%	61%	66%
Rata-rata	69%	69%	69%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	69%	76%	72%
1	72%	66%	69%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	70%	74%	72%
1	72%	68%	70%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	71%	74%	72%
1	72%	68%	70%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	72%	73%	73%
1	72%	71%	72%
Rata-rata	72%	72%	72%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	71%	73%	72%
1	71%	69%	70%
Rata-rata	71%	71%	71%

Tabel 5.37 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan *ADASYN* pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	73%	77%	75%
1	74%	70%	72%
Rata-rata	73%	73%	73%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	74%	77%	76%
1	75%	72%	74%
Rata-rata	75%	75%	75%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	75%	76%	75%
1	75%	73%	74%
Rata-rata	75%	75%	75%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	75%	74%	75%
1	73%	75%	74%
Rata-rata	74%	74%	74%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	75%	74%	75%
1	74%	74%	74%
Rata-rata	74%	74%	74%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	74%	73%	73%
1	73%	74%	73%
Rata-rata	73%	73%	73%
Jumlah *channel* sebanyak 8			

Kelas	*Precision*	*Recall*	*F1-score*
0	72%	70%	71%
1	70%	72%	71%
Rata-rata	71%	71%	71%

Tabel 5.38 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5 dengan *ADASYN*

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	50.19	39.66
2.	40	58.41	37.31
3.	33	57.08	40.28
4.	26	76.28	34.46
5.	19	51.84	36.64
6.	13	34.69	31.18
7.	8	30.14	29.65

5.4.5.3. Dengan Metode *Balancing* Duplikasi

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 2, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.39. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan duplikasi dari kedua subjek dapat dilihat pada Tabel 5.40 dan 5.41. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.42. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.39 Waktu Eksekusi Proses *Balancing* dengan Duplikasi untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	170.51	165.15
2.	40	148.64	154.59
3.	33	135.40	127.28
4.	26	104.41	111.52
5.	19	76.06	70.18
6.	13	58.83	57.61
7.	8	40.47	39.99

Tabel 5.40 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan Duplikasi pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	76%	91%	83%
1	88%	72%	79%
Rata-rata	82%	81%	81%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	76%	90%	83%
1	88%	72%	79%
Rata-rata	82%	81%	81%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	77%	89%	82%
1	87%	73%	80%
Rata-rata	82%	81%	81%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	78%	87%	82%

1	85%	75%	80%
Rata-rata	81%	81%	81%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	81%	85%	83%
1	84%	80%	82%
Rata-rata	82%	82%	82%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	82%	84%	83%
1	84%	81%	82%
Rata-rata	83%	83%	83%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	80%	82%	81%
1	82%	80%	81%
Rata-rata	81%	81%	81%

Tabel 5.41 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 5 dengan Duplikasi pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	86%	90%	88%
1	89%	85%	87%
Rata-rata	88%	88%	88%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	85%	89%	87%
1	89%	85%	87%
Rata-rata	87%	87%	87%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	85%	89%	87%

1	88%	84%	86%
Rata-rata	86%	86%	86%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	82%	87%	85%
1	86%	81%	84%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	83%	83%	83%
1	83%	83%	83%
Rata-rata	83%	83%	83%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	80%	82%
1	81%	85%	83%
Rata-rata	83%	82%	82%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	80%	79%	80%
1	79%	81%	80%
Rata-rata	80%	80%	80%

Tabel 5.42 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 5 dengan Duplikasi

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	13.07	8.87
2.	40	11.46	7.12
3.	33	16.82	8.32
4.	26	23.12	12.78
5.	19	29.24	24.22

6.	13	31.49	35.55
7.	8	29.05	29.74

5.4.6. Skenario Uji Coba 6

Pada skenario uji coba 5, dilakukan proses *balancing* setiap data *training* dan *testing* hasil reduksi dimensi *PCA* dan *LDA*. Kemudian dilanjutkan dengan pembuatan *model*, dan klasifikasi. Berbagai metode *balancing* antar kelas akan dilakukan pada subbab 5.4.6.1 sampai 5.4.6.3.

5.4.6.1. Dengan Metode *Balancing Borderline SMOTE*

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 3, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.43. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan *Borderline SMOTE* dari kedua subjek dapat dilihat pada Tabel 5.44 dan 5.45. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.46. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.43 Waktu Eksekusi Proses *Balancing* dengan *Borderline SMOTE* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.04	0.06
2.	40	0.06	0.07
3.	33	0.07	0.06
4.	26	0.06	0.06
5.	19	0.07	0.06
6.	13	0.04	0.07
7.	8	0.06	0.08

Tabel 5.44 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan *Borderline SMOTE* pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	89%	72%	80%
1	77%	91%	83%
Rata-rata	83%	82%	81%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	76%	81%
1	79%	89%	83%
Rata-rata	83%	82%	82%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	88%	78%	82%
1	80%	89%	84%
Rata-rata	84%	83%	83%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	85%	76%	80%

1	78%	86%	82%
Rata-rata	82%	81%	81%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	75%	81%
1	78%	89%	83%
Rata-rata	83%	82%	82%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	89%	74%	81%
1	78%	91%	84%
Rata-rata	83%	82%	82%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	85%	75%	80%
1	78%	87%	82%
Rata-rata	81%	81%	81%

Tabel 5.45 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan *Borderline SMOTE* pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	97%	80%	88%
1	83%	98%	90%
Rata-rata	90%	89%	89%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	94%	78%	85%
1	81%	95%	88%
Rata-rata	88%	87%	87%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	96%	76%	85%

1	80%	97%	88%
Rata-rata	88%	86%	86%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	92%	75%	83%
1	79%	94%	86%
Rata-rata	85%	84%	84%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	92%	76%	84%
1	80%	94%	86%
Rata-rata	86%	85%	85%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	91%	74%	81%
1	78%	93%	85%
Rata-rata	84%	83%	83%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	90%	70%	79%
1	75%	92%	83%
Rata-rata	83%	81%	81%

Tabel 5.46 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6 dengan *Borderline SMOTE*

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.15	0.12
2.	40	0.16	0.12
3.	33	0.15	0.15
4.	26	0.19	0.18
5.	19	0.19	0.17

6.	13	0.21	0.18
7.	8	0.22	0.25

5.4.6.2. Dengan Metode *Balancing ADASYN*

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 3, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.47. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek dapat dilihat pada Tabel 5.48. Jumlah anggota per kelas dapat berbeda-beda pada setiap data karena dilakukan perhitungan jumlah data sintetis yang akan dibuat menggunakan rasio densitas. Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan *ADASYN* dari kedua subjek dapat dilihat pada Tabel 5.49 dan 5.50. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.51. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.47 Waktu Eksekusi Proses *Balancing* dengan *ADASYN* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.06	0.07
2.	40	0.06	0.08
3.	33	0.09	0.10
4.	26	0.07	0.07
5.	19	0.09	0.07
6.	13	0.07	0.08
7.	8	0.07	0.06

Tabel 5.48 Jumlah Data Per Kelas untuk Skenario Uji Coba 6 dengan *ADASYN*

No	Data	Jumlah *channel*	Jumlah data per kelas	
			Non-target	*Target*
1.	*Training* subjek A	46	2.550	2.537
2.	*Testing* subjek A		3.000	2.988
3.	*Training* subjek B		2.550	2.543
4.	*Testing* subjek B		3.000	2.989
5.	*Training* subjek A	40	2.550	2.547
6.	*Testing* subjek A		3.000	3.071
7.	*Training* subjek B		2.550	2.538
8.	*Testing* subjek B		3.000	2.954
9.	*Training* subjek A	33	2.550	2.555
10.	*Testing* subjek A		3.000	2.991
11.	*Training* subjek B		2.550	2.541
12.	*Testing* subjek B		3.000	2.959
13.	*Training* subjek A	26	2.550	2.564
14.	*Testing* subjek A		3.000	3.022
15.	*Training* subjek B		2.550	2.566

16.	*Testing* subjek B		3.000	3.012
17.	*Training* subjek A	19	2.550	2.575
18.	*Testing* subjek A		3.000	2.988
19.	*Training* subjek B		2.550	2.513
20.	*Testing* subjek B		3.000	3.036
21.	*Training* subjek A	13	2.550	2.550
22.	*Testing* subjek A		3.000	3.026
23.	*Training* subjek B		2.550	2.563
24.	*Testing* subjek B		3.000	3.034
25.	*Training* subjek A	8	2.550	2.538
26.	*Testing* subjek A		3.000	2.999
27.	*Training* subjek B		2.550	2.535
28.	*Testing* subjek B		3.000	3.044

Tabel 5.49 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan *ADASYN* pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	73%	64%	68%
1	68%	76%	72%
Rata-rata	70%	70%	70%

Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	73%	65%	69%
1	69%	76%	72%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	71%	67%	69%
1	68%	73%	70%
Rata-rata	70%	70%	70%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	73%	66%	69%
1	69%	75%	72%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	71%	66%	68%
1	68%	72%	70%
Rata-rata	69%	69%	69%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	69%	65%	67%
1	67%	71%	69%
Rata-rata	68%	68%	68%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	69%	68%	68%
1	68%	69%	69%
Rata-rata	68%	68%	68%

Tabel 5.50 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan *ADASYN* pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	74%	67%	70%
1	70%	76%	73%
Rata-rata	72%	71%	71%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	73%	67%	70%
1	69%	75%	72%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	74%	66%	70%
1	69%	76%	73%
Rata-rata	72%	71%	71%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	72%	67%	70%
1	69%	74%	72%
Rata-rata	71%	71%	71%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	71%	67%	69%
1	69%	73%	71%
Rata-rata	70%	70%	70%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	70%	65%	68%
1	68%	73%	70%
Rata-rata	69%	69%	69%
Jumlah *channel* sebanyak 8			

Kelas	*Precision*	*Recall*	*F1-score*
0	69%	65%	67%
1	67%	72%	69%
Rata-rata	68%	68%	68%

Tabel 5.51 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6 dengan *ADASYN*

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.30	0.25
2.	40	0.31	0.28
3.	33	0.31	0.31
4.	26	0.32	0.32
5.	19	0.40	0.43
6.	13	0.37	0.46
7.	8	0.40	0.39

5.4.6.3. Dengan Metode *Balancing* Duplikasi

Untuk setiap data sinyal dengan variasi jumlah *channel* yang dipertahankan seperti pada skenario uji coba 3, diaplikasikan proses *balancing* dengan waktu eksekusi yang dapat dilihat pada Tabel 5.52. Hasil akhir dari proses *balancing* untuk data dari masing-masing subjek, yaitu data *training* dengan 2.550 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Untuk data *testing*, terdapat 3.000 data dari masing-masing kelas *non-target* (0) dan kelas *target* (1). Evaluasi performa model untuk mengklasifikasi sinyal setelah *balancing* dengan duplikasi dari kedua subjek dapat dilihat pada Tabel 5.53 dan 5.54. Waktu yang dibutuhkan untuk proses *training* dalam pembuatan model pada skenario ini dapat dilihat pada Tabel 5.55. *Confusion matrix* untuk proses klasfikasi pada skenario ini dapat dilihat pada lampiran.

Tabel 5.52 Waktu Eksekusi Proses *Balancing* dengan Duplikasi untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	14.54	13.17
2.	40	13.40	13.20
3.	33	13.32	13.04
4.	26	13.82	12.79
5.	19	13.02	11.07
6.	13	13.27	12.30
7.	8	13.22	11.01

Tabel 5.53 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan Duplikasi pada Subjek A

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	80%	86%	83%
1	85%	78%	81%
Rata-rata	82%	82%	82%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	81%	87%	84%
1	86%	80%	83%
Rata-rata	84%	83%	83%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	83%	85%	84%
1	85%	82%	84%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	85%	85%

1	85%	83%	84%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	83%	85%	84%
1	84%	83%	83%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	84%	84%	84%
1	84%	84%	84%
Rata-rata	84%	84%	84%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	82%	83%	82%
1	82%	81%	82%
Rata-rata	82%	82%	82%

Tabel 5.54 Hasil Evaluasi Performa *Model* untuk Skenario Uji Coba 6 dengan Duplikasi pada Subjek B

Jumlah *channel* sebanyak 46			
Kelas	*Precision*	*Recall*	*F1-score*
0	91%	84%	87%
1	85%	92%	88%
Rata-rata	88%	88%	88%
Jumlah *channel* sebanyak 40			
Kelas	*Precision*	*Recall*	*F1-score*
0	91%	85%	88%
1	86%	92%	89%
Rata-rata	89%	89%	88%
Jumlah *channel* sebanyak 33			
Kelas	*Precision*	*Recall*	*F1-score*
0	90%	87%	88%

1	87%	90%	89%
Rata-rata	88%	88%	88%
Jumlah *channel* sebanyak 26			
Kelas	*Precision*	*Recall*	*F1-score*
0	89%	85%	87%
1	86%	89%	87%
Rata-rata	87%	87%	87%
Jumlah *channel* sebanyak 19			
Kelas	*Precision*	*Recall*	*F1-score*
0	88%	82%	85%
1	83%	88%	86%
Rata-rata	85%	85%	85%
Jumlah *channel* sebanyak 13			
Kelas	*Precision*	*Recall*	*F1-score*
0	87%	82%	84%
1	83%	88%	85%
Rata-rata	85%	85%	85%
Jumlah *channel* sebanyak 8			
Kelas	*Precision*	*Recall*	*F1-score*
0	83%	80%	81%
1	80%	83%	82%
Rata-rata	82%	81%	81%

Tabel 5.55 Waktu Eksekusi Proses *Training* untuk Setiap Variasi Jumlah *Channel* yang Dipertahankan pada Skenario Uji Coba 6 dengan Duplikasi

No	Jumlah *channel*	Waktu eksekusi (dalam detik)	
		Subjek A	Subjek B
1.	46	0.03	0.01
2.	40	0.01	0.01
3.	33	0.03	0.03
4.	26	0.04	0.03
5.	19	0.04	0.06

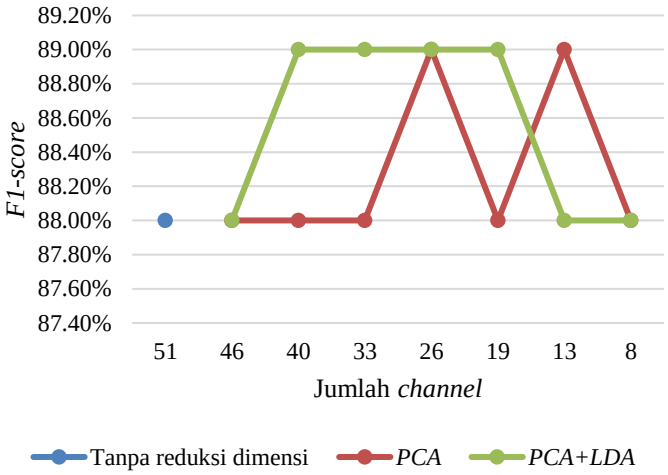
6.	13	0.07	0.08
7.	8	0.14	0.13

5.5. Analisis Hasil Uji Coba

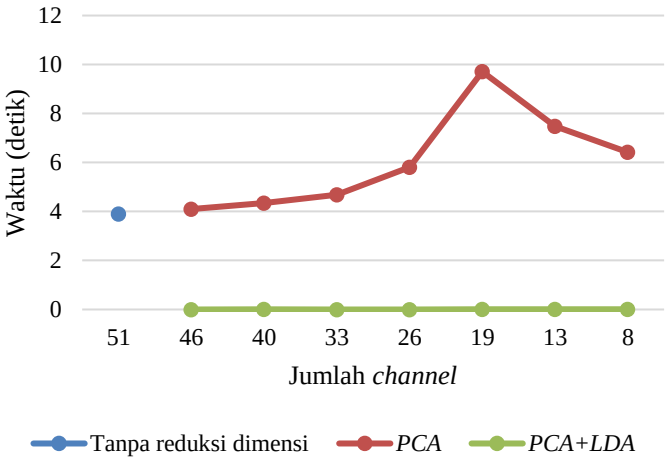
Pada skenario uji coba 1, dilakukan klasifikasi sinyal yang sudah melewati proses *preprocessing* sinyal, namun tidak mengalami reduksi dimensi dan proses *balancing*. Oleh karena itu, skenario ini dijadikan skenario dasar yang menjadi pembanding dengan skenario-skenario lainnya yang menerapkan proses reduksi dimensi dan *balancing*. Pada skenario uji coba 1, diperoleh *f1-score* rata-rata sebesar 88% untuk subjek A dan 90% untuk subjek B. Untuk masing-masing subjek A dan B, diperlukan waktu *training* selama 3,9 detik dan 3,07 detik.

Pada skenario uji coba 2, dilakukan reduksi dimensi menggunakan *PCA*. Hasil *f1-score* rata-rata terbaik untuk subjek A adalah data dengan reduksi dimensi yang mempertahankan 13 *channel*, yaitu 89% dengan waktu *training* 7,48 detik. Hasil *f1-score* rata-rata terbaik untuk subjek B adalah data dengan reduksi dimensi yang mempertahankan 46 *channel*, yaitu 90% dengan waktu *training* 2,92 detik.

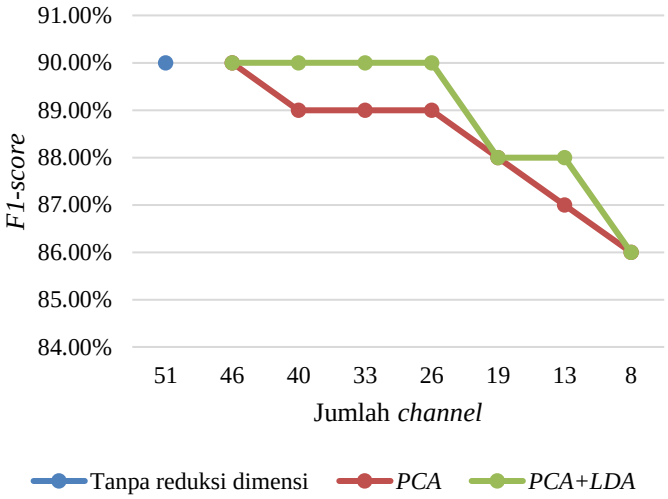
Pada skenario uji coba 3, data yang sudah melewati proses reduksi dimensi menggunakan *PCA*, akan mengalami reduksi dimensi temporal menggunakan *LDA*. Hasil *f1-score* rata-rata terbaik untuk subjek A adalah data dengan reduksi dimensi yang mempertahankan 26 *channel*, yaitu 89%. Hasil *f1-score* rata-rata terbaik untuk subjek B adalah data dengan reduksi dimensi yang mempertahankan 26 *channel*, yaitu 90%. Waktu *training* untuk kedua subjek selesai secara instan sehingga tidak terekam oleh pengukur waktu eksekusi. Grafik hubungan antara jumlah *channel* yang dipertahankan dengan *f1-score* rata-rata dapat dilihat pada Gambar 5.7 dan 5.8



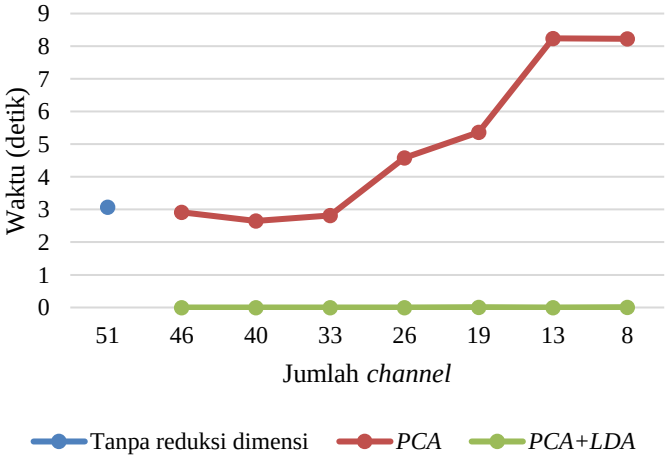
Gambar 5.7 Grafik hubungan *f1-score* dengan metode reduksi dimensi pada skenario uji coba 1 sampai 3 untuk subjek A



Gambar 5.8 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 1 sampai 3 untuk subjek A

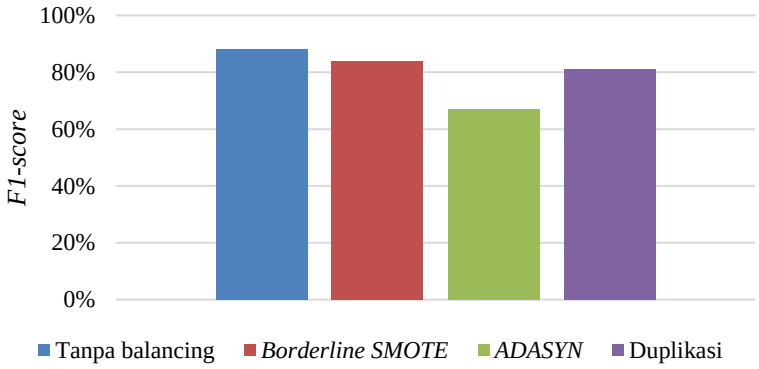


Gambar 5.9 Grafik hubungan *f1-score* dengan metode reduksi dimensi pada skenario uji coba 1 sampai 3 untuk subjek B

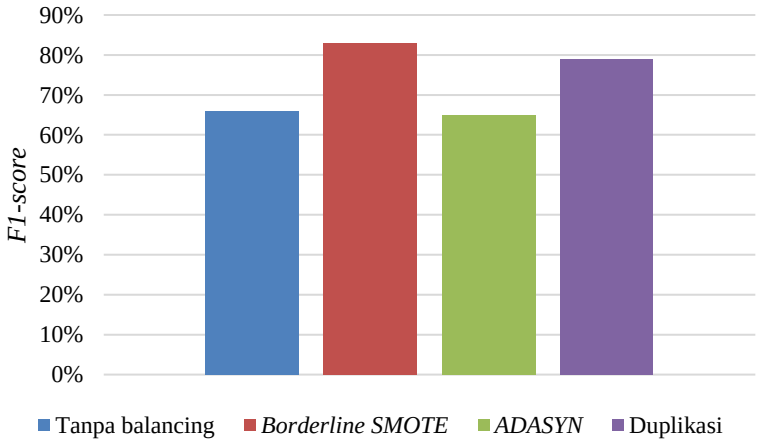


Gambar 5.10 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 1 sampai 3 untuk subjek B

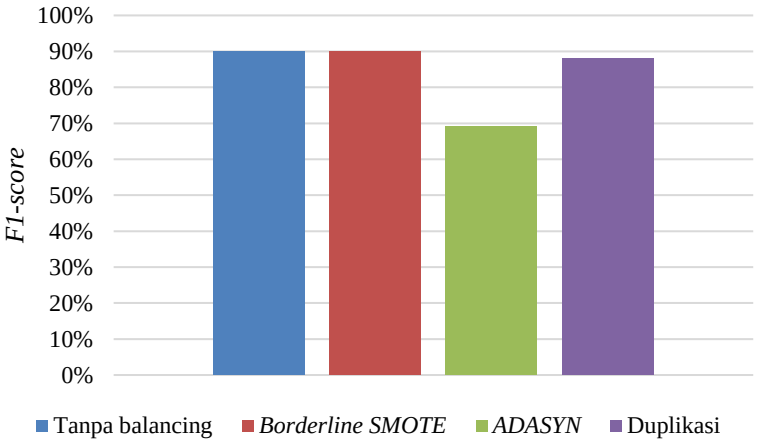
Pada skenario uji coba 4, dilakukan proses *balancing* pada data *training* dan *testing* untuk meningkatkan *f1-score* pada kelas *target* (1). Proses klasifikasi data yang sudah melewati proses *balancing* dengan *Borderline SMOTE* menghasilkan *f1-score* rata-rata sebesar 84% untuk subjek A dan 90% untuk subjek B. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 66% menjadi 83%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 73% menjadi 90%. Untuk data yang melewati proses *balancing* dengan *ADASYN* menghasilkan *f1-score* rata-rata sebesar 67% untuk subjek A dan 69% untuk subjek B. Pada subjek A, terdapat penurunan *f1-score* kelas *target* (1) dari 66% menjadi 65%. Pada subjek B, terdapat penurunan *f1-score* kelas *target* (1) dari 73% menjadi 67%. Data yang melewati proses *balancing* dengan duplikasi menghasilkan *f1-score* rata-rata sebesar 81% untuk subjek A dan 88% untuk subjek B. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 66% menjadi 79%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 73% menjadi 87%. Grafik hubungan metode *balancing* dengan *f1-score* dapat dilihat pada Gambar 5.11 sampai 5.14. Pada skenario uji coba ini, metode *balancing* yang dapat memberikan performa klasifikasi terbaik untuk subjek A dan B adalah *Borderline SMOTE*.



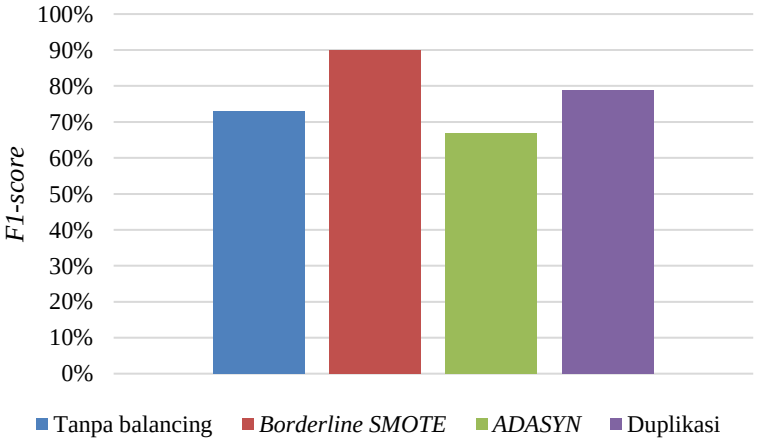
Gambar 5.11 Grafik hubungan *f1-score* rata-rata dengan metode *balancing* pada skenario uji coba 4 untuk subjek A



Gambar 5.12 Grafik hubungan *f1-score* kelas *target* (1) dengan metode *balancing* pada skenario uji coba 4 untuk subjek A

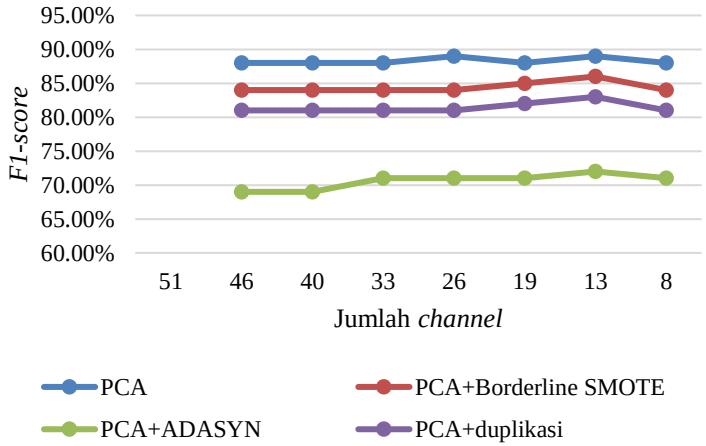


Gambar 5.13 Grafik hubungan *f1-score* rata-rata dengan metode *balancing* pada skenario uji coba 4 untuk subjek B

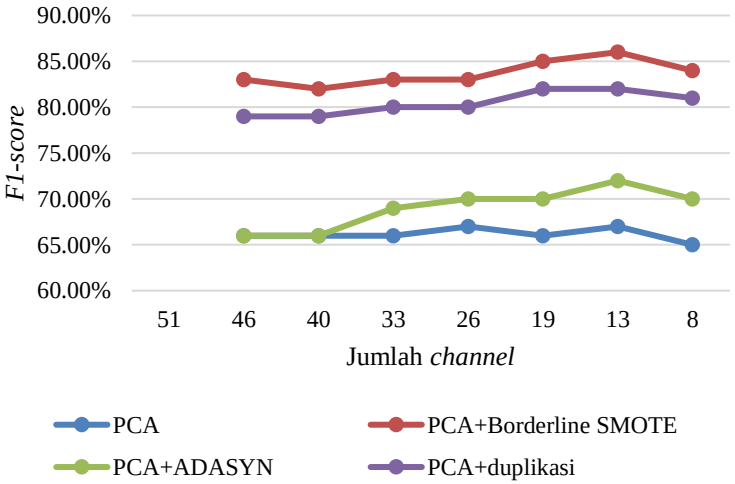


Gambar 5.14 Grafik hubungan *f1-score* kelas *target* (1) dengan metode *balancing* pada skenario uji coba 4 untuk subjek B

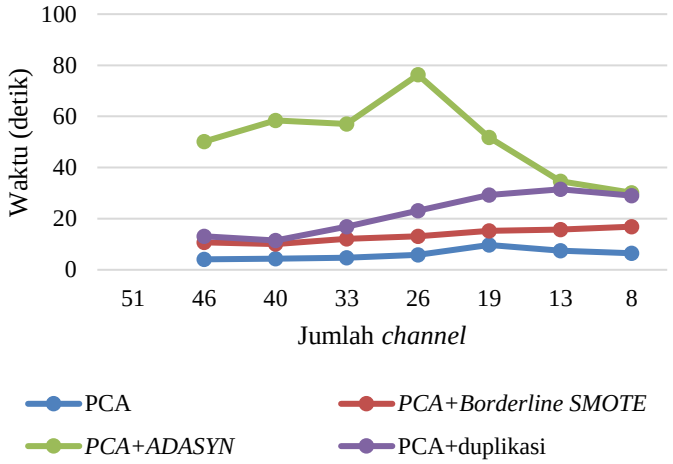
Pada skenario uji coba 5, dilakukan proses *balancing* pada data *training* dan *testing* yang sudah melewati proses reduksi dimensi *PCA*. Proses klasifikasi data yang sudah melewati proses *balancing* dengan *Borderline SMOTE* menghasilkan *f1-score* rata-rata terbaik sebesar 86% untuk subjek A dengan jumlah *channel* sebanyak 13 *channel*, dan 90% untuk subjek B dengan jumlah *channel* sebanyak 40 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 67% menjadi 86%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 71% menjadi 90%. Untuk data yang melewati proses *balancing* dengan *ADASYN*, dihasilkan *f1-score* rata-rata terbaik sebesar 72% untuk subjek A jumlah *channel* sebanyak 13 *channel*, dan 75% untuk subjek B dengan jumlah *channel* sebanyak 40 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 67% menjadi 72%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 71% menjadi 74%. Data yang melewati proses *balancing* dengan duplikasi menghasilkan *f1-score* rata-rata terbaik sebesar 83% untuk subjek A dengan jumlah *channel* sebanyak 13 *channel*, dan 88% untuk subjek B dengan jumlah *channel* sebanyak 46 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 67% menjadi 82%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 72% menjadi 87%. Grafik hubungan metode *balancing* dan reduksi dimensi *PCA* dengan *f1-score* dapa dilihat pada Gambar 5.15 sampai 5.18. Pada skenario uji coba ini, metode *balancing* yang dapat memberikan performa klasifikasi terbaik untuk subjek A dan B adalah *Borderline SMOTE*.



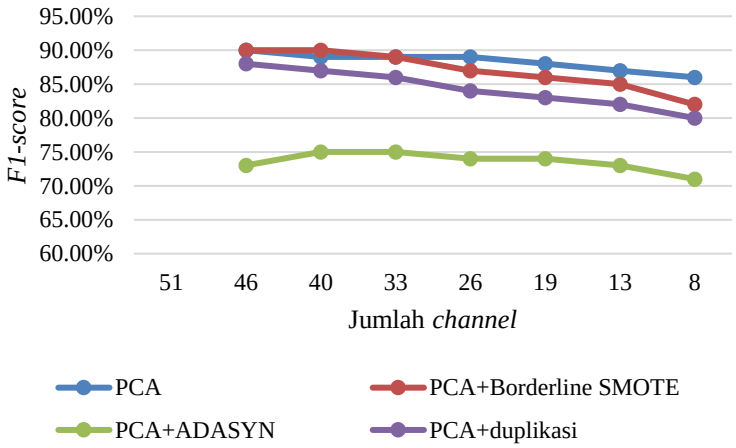
Gambar 5.15 Grafik hubungan *f1-score* dengan metode reduksi dimensi *PCA* dan *balancing* pada skenario uji coba 5 untuk subjek A



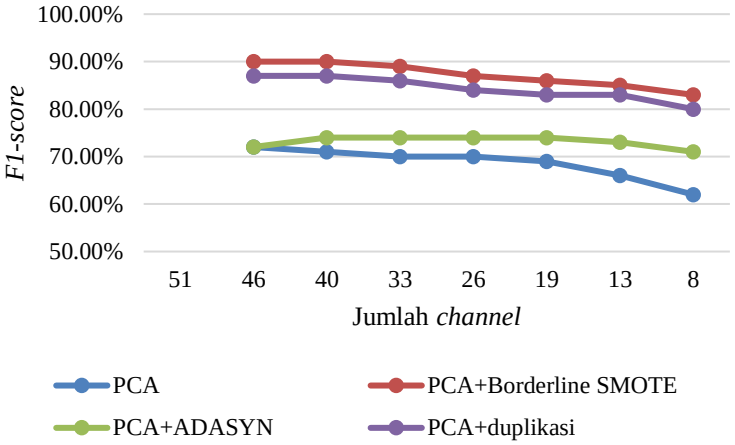
Gambar 5.16 Grafik hubungan *f1-score* kelas *target* (1) dengan metode reduksi dimensi *PCA* dan *balancing* pada skenario uji coba 5 untuk subjek A



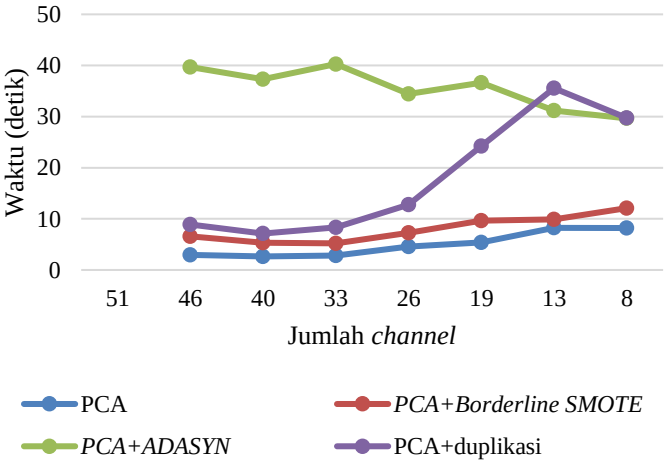
Gambar 5.17 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 5 untuk subjek A



Gambar 5.18 Grafik hubungan *f1-score* dengan metode reduksi dimensi *PCA* dan *balancing* pada skenario uji coba 5 untuk subjek B

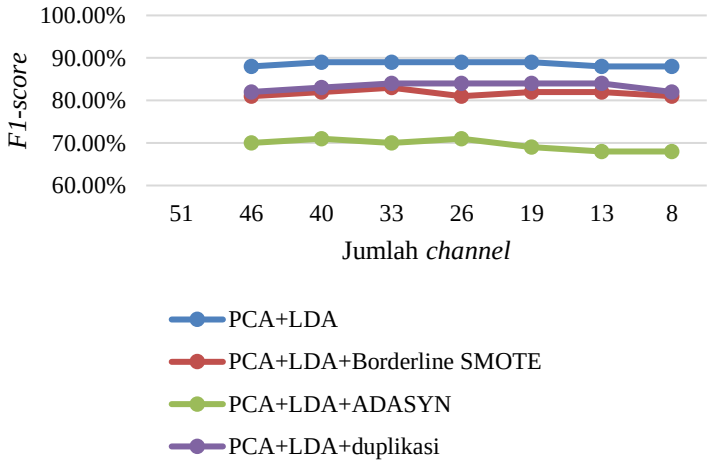


Gambar 5.19 Grafik hubungan *f1-score* kelas *target* (1) dengan metode reduksi dimensi *PCA* dan *balancing* pada skenario uji coba 5 untuk subjek B

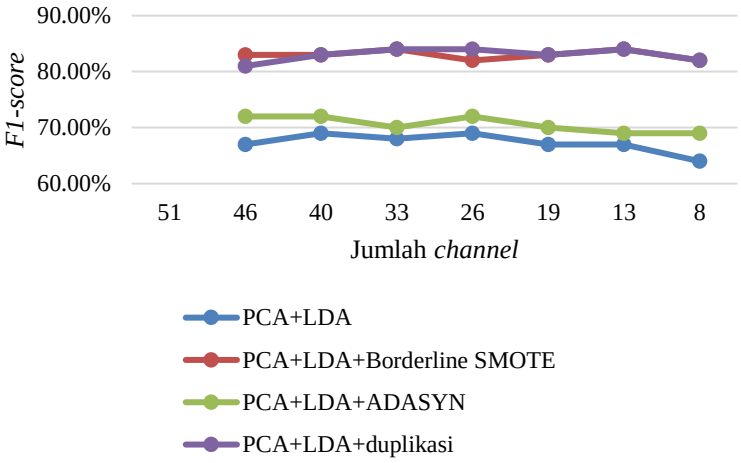


Gambar 5.20 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 5 untuk subjek B

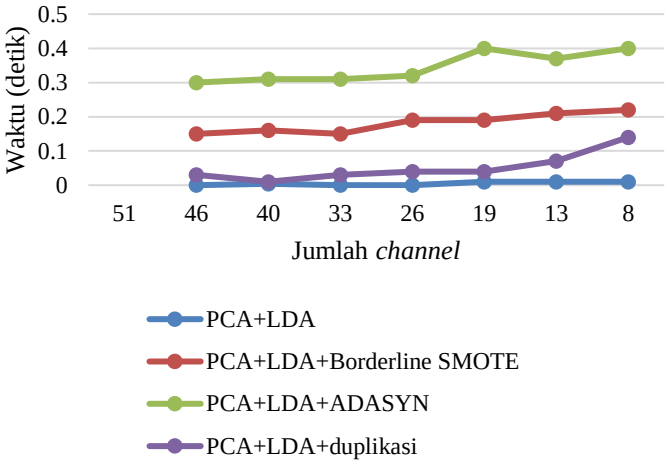
Pada skenario uji coba 6, dilakukan proses *balancing* pada data *training* dan *testing* yang sudah melewati proses reduksi dimensi *PCA* dan *LDA*. Proses klasifikasi data yang sudah melewati proses *balancing* dengan *Borderline SMOTE* menghasilkan *f1-score* rata-rata terbaik sebesar 83% untuk subjek A dengan jumlah *channel* sebanyak 33 *channel*, dan 89% untuk subjek B dengan jumlah *channel* sebanyak 46 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 68% menjadi 84%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 73% menjadi 90%. Untuk data yang melewati proses *balancing* dengan *ADASYN*, dihasilkan *f1-score* rata-rata terbaik sebesar 71% untuk subjek A dengan jumlah *channel* sebanyak 26 *channel*, dan 71% untuk subjek B dengan jumlah *channel* sebanyak 46 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 69% menjadi 72%. Pada subjek B, tidak terjadi perubahan *f1-score* kelas *target* (1). Data yang melewati proses *balancing* dengan duplikasi menghasilkan *f1-score* rata-rata terbaik sebesar 84% untuk subjek A dengan jumlah *channel* sebanyak 26 *channel*, dan 88% untuk subjek B dengan jumlah *channel* sebanyak 40 *channel*. Pada subjek A, terdapat peningkatan *f1-score* kelas *target* (1) dari 69% menjadi 84%. Pada subjek B, terdapat peningkatan *f1-score* kelas *target* (1) dari 73% menjadi 89%. Pada skenario uji coba ini, metode *balancing* yang dapat memberikan performa klasifikasi terbaik untuk subjek A adalah dengan duplikasi, dan subjek B dengan *Borderline SMOTE*.



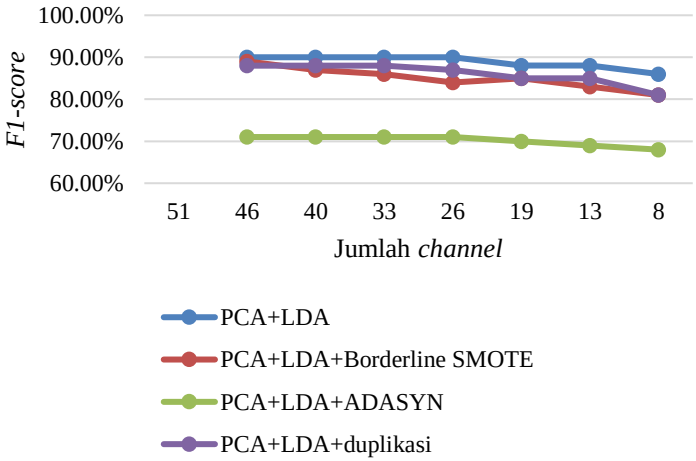
Gambar 5.21 Grafik hubungan *f1-score* dengan metode reduksi dimensi *PCA*, *LDA*, dan *balancing* pada skenario uji coba 6 untuk subjek A



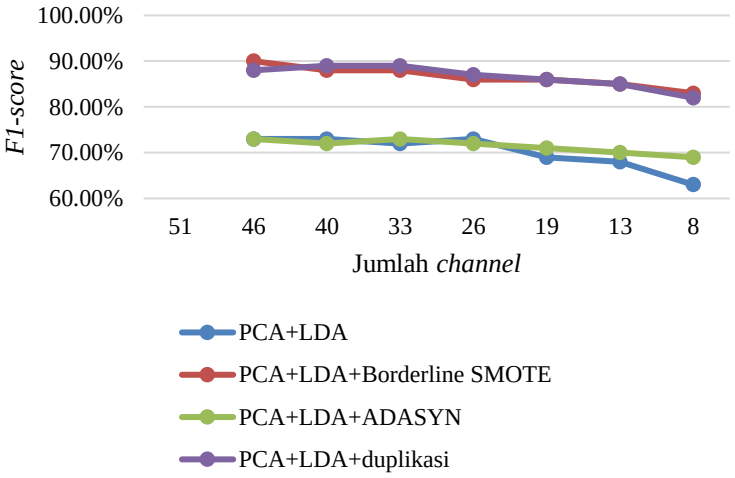
Gambar 5.22 Grafik hubungan *f1-score* kelas *target* (1) dengan metode reduksi dimensi *PCA*, *LDA*, dan *balancing* pada skenario uji coba 6 untuk subjek A



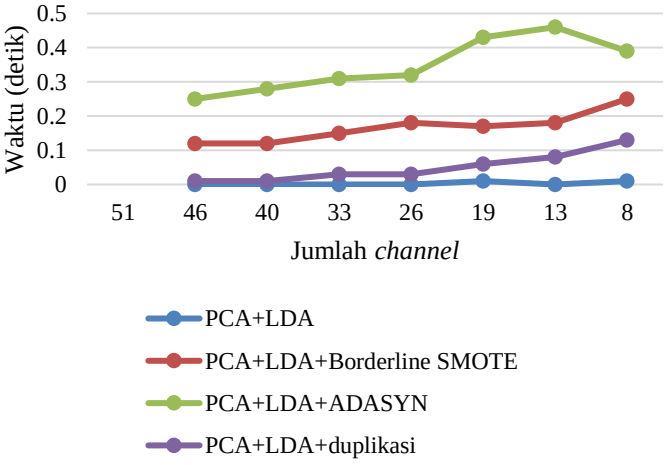
Gambar 5.23 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 6 untuk subjek A



Gambar 5.24 Grafik hubungan *f1-score* dengan metode reduksi dimensi *PCA, LDA*, dan *balancing* pada skenario uji coba 6 untuk subjek B



Gambar 5.25 Grafik hubungan *f1-score* kelas *target* (1) dengan metode reduksi dimensi *PCA*, *LDA*, dan *balancing* pada skenario uji coba 6 untuk subjek B



Gambar 5.26 Grafik hubungan waktu *training* dengan metode reduksi dimensi pada skenario uji coba 6 untuk subjek B

Dari skenario uji coba 4 sampai 6, metode *balancing ADASYN* secara konsisten memberikan performa klasifikasi yang lebih buruk dibandingkan metode *balancing* lainnya. Metode terbaik untuk mengklasifikasi sinyal EEG dari subjek A adalah dengan mengaplikasikan reduksi dimensi *PCA* dengan jumlah *channel* sebanyak 13 *channel* dan *balancing* dengan *Borderline SMOTE*. Metode ini dapat mengklasifikasi sinyal EEG dari subjek A dengan *f1-score* kedua kelas konsisten pada angka 86%. Untuk subjek B, metode terbaik adalah dengan mengaplikasikan metode *balancing Borderline SMOTE* tanpa reduksi dimensi. Metode ini dapat mengklasifikasi sinyal EEG dari subjek B dengan *f1-score* kedua kelas di angka 90%.

Proses reduksi dimensi *LDA* dapat mengurangi waktu yang dibutuhkan untuk *training* hingga dibawah 1 detik. Untuk subjek A, performa klasifikasi terbaik dicapai dengan melakukan reduksi dimensi *PCA* yang mempertahankan 26 *channel*, kemudian *LDA*, dan *balancing* dengan duplikasi. Metode ini berhasil mengklasifikasi sinyal EEG subjek A dengan *f1-score* kedua kelas sebesar 84% dengan durasi proses *training* 0,04 detik. *F1-score* dari metode ini lebih rendah sebesar 2% dari metode terbaik yang dihasilkan dari seluruh skenario uji coba, yang membutuhkan waktu *training* selama 15,77 detik. Untuk subjek B, performa klasifikasi terbaik dicapai dengan melakukan reduksi dimensi *PCA* yang mempertahankan 46 *channel*, kemudian *LDA*, dan *balancing* dengan *Borderline SMOTE*. Metode ini berhasil mengklasifikasi sinyal EEG subjek B dengan *f1-score* kedua kelas sebesar 89% dengan durasi proses *training* 0,12 detik. *F1-score* dari metode ini lebih rendah sebesar 1% dari metode terbaik yang dihasilkan dari seluruh skenario uji coba, yang membutuhkan waktu *training* selama 6,64 detik.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

6.1. Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Metode pemrosesan data EEG terbaik untuk mengklasifikasikan sinyal EEG dari subjek A adalah melakukan reduksi dimensi *PCA* dengan mempertahankan jumlah *channel* sebanyak 13 *channel* dan *balancing* dengan *Borderline SMOTE*, yang menghasilkan *f1-score* 86% untuk kedua kelas. Waktu *training* yang dibutuhkan adalah 15,77 detik.
2. Metode pemrosesan data EEG terbaik terbaik untuk mengklasifikasikan sinyal EEG dari subjek B adalah dengan melakukan *balancing* dengan *Borderline SMOTE* tanpa reduksi dimensi, yang menghasilkan *f1-score* 90% untuk kedua kelas. Waktu *training* yang dibutuhkan adalah 6,64 detik.
3. Pada skenario uji coba 4, proses *balancing* dengan *ADASYN* mengakibatkan penurunan *f1-score* dari kelas *target* (1). Selain itu, proses *balancing* dengan *Borderline SMOTE*, *ADASYN*, dan duplikasi secara konsisten meningkatkan *f1-score* dari kelas *target* (1).
4. Pada subjek A, proses eliminasi *channel* dengan *PCA* dapat meningkatkan performa klasifikasi.
5. Proses eliminasi *channel* dengan *PCA* tidak mempersingkat waktu *training*. Akan tetapi, reduksi dimensi temporal dan *channel* dengan *LDA* berhasil mempersingkat waktu *training*

hingga dibawah 1 detik untuk seluruh skenario uji coba yang melibatkan proses reduksi dimensi *LDA*.

6. Metode terbaik yang melibatkan reduksi dimensi menggunakan *LDA* untuk subjek A adalah dengan melakukan reduksi dimensi *PCA* yang mempertahankan 26 *channel*, kemudian *LDA*, dan *balancing* dengan duplikasi. Metode ini berhasil mengklasifikasi sinyal EEG subjek A dengan *f1-score* kedua kelas sebesar 84% dengan durasi proses *training* 0,04 detik.
7. Metode terbaik yang melibatkan reduksi dimensi menggunakan *LDA* untuk subjek B adalah dengan melakukan reduksi dimensi *PCA* yang mempertahankan 46 *channel*, kemudian *LDA*, dan *balancing* dengan *Borderline SMOTE*. Metode ini berhasil mengklasifikasi sinyal EEG subjek B dengan *f1-score* kedua kelas sebesar 89% dengan durasi proses *training* 0,12 detik

6.2. Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Peningkatan kemampuan perangkat keras untuk melakukan komputasi dapat mempercepat proses *training*, serta memungkinkan penelitian pada banyak parameter.
2. Melakukan eksperimen dengan mengubah parameter-parameter pada proses pembuatan *model* dengan *SVM*.
3. Melakukan klasifikasi dengan metode yang lain.
4. Mencoba sistem ini dengan menggunakan rekaman sinyal EEG *event related potential* lain.

Demikian saran yang dapat menjadi masukan dalam pengembangan selanjutnya.

DAFTAR PUSTAKA

[1] R. Fazel-Rezai, B. Z. Allison, C. Guger, E. W. Sellers, S. C. Kleih dan A. Kübler, "P300 brain computer interface: current challenges and emerging trends," *Frontiers in Neuroengineering*, vol. 5, pp. 1-14, 17 Juli 2012.

[2] BCI Competition III, "Wadsworth BCI Dataset (P300 Evoked Potentials)," 2004. [Online]. Available: http://www.bbci.de/competition/iii/. [Diakses 16 Mei 2018].

[3] H. Tjandrasa dan S. Djanali, "Classification of P300 Event-Related Potentials Using Wavelet Transform, MLP, and Soft Margin SVM," dalam *The 10th International Conference on Advanced Computational Intelligence 2018*, Xiamen, 2018.

[4] M. Thulasidas, S. Cuntai Guan dan J. Wu, "Robust Classification of EEG Signal for Brain–Computer Interface," *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING*, vol. 14, no. 1, pp. 24-29, 2006.

[5] L. D. Paarmann, Design and Analysis of Analog Filters: A Signal Processing Perspective, Kansas: KLUWER ACADEMIC PUBLISHERS , 2001.

[6] W. Weckesser, "Butterworth Bandpass — SciPy Cookbook documentation," 2015. [Online]. Available: https://scipy-cookbook.readthedocs.io/items/ButterworthBandpass.html. [Diakses 14 Desember 2018].

[7] L. Milic, Multirate Filtering for Digital Signal Processing: MATLAB Applications, Belgrade: IGI Global, 2009.

[8] MathWorks, "Decimation — decrease sample rate by integer factor," [Online]. Available: https://www.mathworks.com/help/signal/ref/decimate.html. [Diakses 13 Desember 2018].

[9] P. Tagare, "Signal averaging," dalam *Biomedical digital signal processing*, Willis Tompkins Editor, 1993, pp. 184-192.

[10] Suyanto, Data Mining untuk Klasifikasi dan Klasterisasi Data, Bandung: Penerbit Informatika, 2017.

[11] C. C. Aggarwal, Data Mining The Textbook, New York: Springer, 2015.

[12] H. He, E. A, Y. Bai dan G. S. Li, "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning," *2008 InternationalJoint Conference on Neural Networks (IJCNN 2008)*, pp. 1322-1328, 2008.

[13] H. He, "Learning from Imbalanced Data," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 21, no. 9, pp. 1263-1284, 2009.

[14] A. Rakotomamonjy dan V. Guigue, "BCI Competition III: Dataset II - Ensemble of SVMs for BCI P300 Speller," INSA de Rouen, Saint Etienne du Rouvray, 2008.

[15] A. PURNOMO, H. TJANDRASA dan D. A. NAVASTARA, "DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK," Jurusan Teknik Informatika-ITS., ITS Surabaya, 2018.

[16] K. M. Ong, K. H. Thung, C. Y. Wee dan R. Paramesran, "Selection of a Subset of EEG Channels using PCA to classify Alcoholics and Non-alcoholics," dalam *IEEE Engineering in Medicine and Biology 27th Annual Conference*, Shanghai, 2005.

[17] MathWorks, "Butterworth filter design - MATLAB butter," [Online]. Available: https://www.mathworks.com/help/signal/ref/butter.html. [Diakses 22 Desember 2018].

[18] MathWorks, "Decrease sample rate by integer factor - MATLAB downsample," [Online]. Available:

https://www.mathworks.com/help/signal/ref/downsample.html. [Diakses 22 Desember 2018].

[19] Python Software Foundation, “Data Structures — Python 3.7.2rc1 documentation,” 24 12 2018. [Online]. Available: https://docs.python.org/3/tutorial/datastructures.html#dictionaries. [Diakses 24 Desember 2018].

[20] MathWorks, “Write comma-separated value file - MATLAB csvwrite,” [Online]. Available: https://www.mathworks.com/help/matlab/ref/csvwrite.html. [Diakses 24 Desember 2018].

[21] scikit-learn developers, “sklearn.preprocessing.StandardScaler — scikit-learn 0.20.2 documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. [Diakses 24 Desember 2018].

[22] scikit-learn developers, “sklearn.decomposition.PCA — scikit-learn 0.20.2 documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. [Diakses 24 Desember 2018].

[23] scikit-learn developers, “sklearn.discriminant_analysis.LinearDiscriminantAnalysis — scikit-learn 0.20.2 documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html. [Diakses 24 Desember 2018].

[24] G. Lemaitre, F. Nogueira, D. Oliveira dan C. Aridas, “imblearn.over_sampling.SMOTE — imbalanced-learn 0.3.0.dev0 documentation,” [Online]. Available: http://glemaitre.github.io/imbalanced-learn/generated/imblearn.over_sampling.SMOTE.html#. [Diakses 24 Desember 2018].

[25] G. Lemaitre, F. Nogueira, D. Oliveira dan C. Aridas, "imblearn.over_sampling.ADASYN — imbalanced-learn 0.3.0.dev0 documentation," [Online]. Available: http://glemaitre.github.io/imbalanced-learn/generated/imblearn.over_sampling.ADASYN.html. [Diakses 24 Desember 2018].

[26] scikit-learn developers, "sklearn.svm.SVC — scikit-learn 0.20.2 documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC. [Diakses 25 Desember 2018].

LAMPIRAN

1. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 1 :

```
Train raw SubjectA
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 3.9028005599975586
[[2733  267]
 [ 171  429]]
             precision    recall  f1-score   support

          0       0.94      0.91      0.93      3000
          1       0.62      0.71      0.66       600

avg / total       0.89      0.88      0.88      3600

Train raw SubjectB
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 3.0774388313293457
[[2704  296]
 [  89  511]]
             precision    recall  f1-score   support

          0       0.97      0.90      0.93      3000
          1       0.63      0.85      0.73       600

avg / total       0.91      0.89      0.90      3600
```

2. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 2:

```
Train SubjectA pca 46 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 4.109754800796509
[[2722  278]
 [ 169  431]]
             precision    recall  f1-score   support
```

```
          0       0.94      0.91      0.92      3000
          1       0.61      0.72      0.66       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA pca 40 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 4.343637704849243
[[2747  253]
 [ 179  421]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.62      0.70      0.66       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA pca 33 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 4.684446096420288
[[2748  252]
 [ 176  424]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.63      0.71      0.66       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA pca 26 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 5.81078839302063
[[2765  235]
 [ 180  420]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.64      0.70      0.67       600

avg / total       0.89      0.88      0.89      3600
```

```
Train SubjectA pca 19 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 9.718732357025146
[[2749  251]
 [ 184  416]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.62      0.69      0.66       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA pca 13 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 7.480717658996582
[[2771  229]
 [ 181  419]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.65      0.70      0.67       600

avg / total       0.89      0.89      0.89      3600

Train SubjectA pca 8 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 6.426708936691284
[[2769  231]
 [ 203  397]]
             precision    recall  f1-score   support

          0       0.93      0.92      0.93      3000
          1       0.63      0.66      0.65       600

avg / total       0.88      0.88      0.88      3600

Train SubjectB pca 46 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
```

```
Training time: 2.92126727104187
[[2700  300]
 [  89  511]]
             precision    recall  f1-score   support

          0       0.97      0.90      0.93      3000
          1       0.63      0.85      0.72       600

avg / total       0.91      0.89      0.90      3600

Train SubjectB pca 40 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 2.657417058944702
[[2684  316]
 [  95  505]]
             precision    recall  f1-score   support

          0       0.97      0.89      0.93      3000
          1       0.62      0.84      0.71       600

avg / total       0.91      0.89      0.89      3600

Train SubjectB pca 33 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 2.8232078552246094
[[2666  334]
 [  98  502]]
             precision    recall  f1-score   support

          0       0.96      0.89      0.93      3000
          1       0.60      0.84      0.70       600

avg / total       0.90      0.88      0.89      3600

Train SubjectB pca 26 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 4.588474273681641
[[2690  310]
 [ 111  489]]
             precision    recall  f1-score   support
```

```
          0       0.96      0.90      0.93      3000
          1       0.61      0.81      0.70       600

avg / total       0.90      0.88      0.89      3600

Train SubjectB pca 19 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 5.3731420040130615
[[2653  347]
 [ 106  494]]
             precision    recall  f1-score   support

          0       0.96      0.88      0.92      3000
          1       0.59      0.82      0.69       600

avg / total       0.90      0.87      0.88      3600

Train SubjectB pca 13 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 8.248550653457642
[[2654  346]
 [ 133  467]]
             precision    recall  f1-score   support

          0       0.95      0.88      0.92      3000
          1       0.57      0.78      0.66       600

avg / total       0.89      0.87      0.87      3600

Train SubjectB pca 8 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 8.237439393997192
[[2644  356]
 [ 171  429]]
             precision    recall  f1-score   support

          0       0.94      0.88      0.91      3000
          1       0.55      0.71      0.62       600

avg / total       0.87      0.85      0.86      3600
```

3. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 3:

```
Train SubjectA lda 46 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2722  278]
 [ 159  441]]
             precision    recall  f1-score   support

          0       0.94      0.91      0.93      3000
          1       0.61      0.73      0.67       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA lda 40 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.004986286163330078
[[2750  250]
 [ 155  445]]
             precision    recall  f1-score   support

          0       0.95      0.92      0.93      3000
          1       0.64      0.74      0.69       600

avg / total       0.90      0.89      0.89      3600

Train SubjectA lda 33 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2770  230]
 [ 172  428]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.65      0.71      0.68       600

avg / total       0.89      0.89      0.89      3600

Train SubjectA lda 26 channel
target test data =  600
```

```
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2755  245]
 [ 157  443]]
             precision    recall  f1-score   support

          0       0.95      0.92      0.93      3000
          1       0.64      0.74      0.69       600

avg / total       0.90      0.89      0.89      3600

Train SubjectA lda 19 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.015622138977050781
[[2751  249]
 [ 169  431]]
             precision    recall  f1-score   support

          0       0.94      0.92      0.93      3000
          1       0.63      0.72      0.67       600

avg / total       0.89      0.88      0.89      3600

Train SubjectA lda 13 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.015622138977050781
[[2732  268]
 [ 165  435]]
             precision    recall  f1-score   support

          0       0.94      0.91      0.93      3000
          1       0.62      0.72      0.67       600

avg / total       0.89      0.88      0.88      3600

Train SubjectA lda 8 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.015621662139892578
[[2777  223]
```

```
 [ 210  390]]
             precision    recall  f1-score   support

          0       0.93      0.93      0.93      3000
          1       0.64      0.65      0.64       600

avg / total       0.88      0.88      0.88      3600

Train SubjectB lda 46 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2692  308]
 [  74  526]]
             precision    recall  f1-score   support

          0       0.97      0.90      0.93      3000
          1       0.63      0.88      0.73       600

avg / total       0.92      0.89      0.90      3600

Train SubjectB lda 40 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2692  308]
 [  73  527]]
             precision    recall  f1-score   support

          0       0.97      0.90      0.93      3000
          1       0.63      0.88      0.73       600

avg / total       0.92      0.89      0.90      3600

Train SubjectB lda 33 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2701  299]
 [  89  511]]
             precision    recall  f1-score   support

          0       0.97      0.90      0.93      3000
          1       0.63      0.85      0.72       600
```

```
avg / total       0.91      0.89      0.90      3600

Train SubjectB lda 26 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2718  282]
 [  91  509]]
             precision    recall  f1-score   support

          0       0.97      0.91      0.94      3000
          1       0.64      0.85      0.73       600

avg / total       0.91      0.90      0.90      3600

Train SubjectB lda 19 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.015621423721313477
[[2667  333]
 [ 110  490]]
             precision    recall  f1-score   support

          0       0.96      0.89      0.92      3000
          1       0.60      0.82      0.69       600

avg / total       0.90      0.88      0.88      3600

Train SubjectB lda 13 channel
target test data =  600
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.0
[[2664  336]
 [ 115  485]]
             precision    recall  f1-score   support

          0       0.96      0.89      0.92      3000
          1       0.59      0.81      0.68       600

avg / total       0.90      0.87      0.88      3600

Train SubjectB lda 8 channel
target test data =  600
```

```
non target test data =  3000
target train data =  510
non target train data =  2550
Training time: 0.015622138977050781
[[2652  348]
 [ 164  436]]
             precision    recall  f1-score   support

          0       0.94      0.88      0.91      3000
          1       0.56      0.73      0.63       600

avg / total       0.88      0.86      0.86      3600
```

4. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 4:

```
Train SubjectA resampled_borderline
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 9.237013101577759
[[2733  267]
 [ 682 2318]]
             precision    recall  f1-score   support

          0       0.80      0.91      0.85      3000
          1       0.90      0.77      0.83      3000

avg / total       0.85      0.84      0.84      6000

Train SubjectB resampled_borderline
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 6.646510601043701
[[2704  296]
 [ 283 2717]]
             precision    recall  f1-score   support

          0       0.91      0.90      0.90      3000
          1       0.90      0.91      0.90      3000

avg / total       0.90      0.90      0.90      6000

Train SubjectA resampled_adasyn
target test data =  2965
```

```
non target test data =  3000
target train data =  2558
non target train data =  2550
Training time: 2936.9858446121216
[[2255  745]
 [1186 1779]]
             precision    recall  f1-score   support

          0       0.66      0.75      0.70      3000
          1       0.70      0.60      0.65      2965

avg / total       0.68      0.68      0.67      5965

Train SubjectB resampled_adasyn
target test data =  2913
non target test data =  3000
target train data =  2614
non target train data =  2550
Training time: 1122.0140326023102
[[2223  777]
 [1064 1849]]
             precision    recall  f1-score   support

          0       0.68      0.74      0.71      3000
          1       0.70      0.63      0.67      2913

avg / total       0.69      0.69      0.69      5913

Train SubjectA resampled_duplication
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 10.855348587036133
[[2733  267]
 [ 855 2145]]
             precision    recall  f1-score   support

          0       0.76      0.91      0.83      3000
          1       0.89      0.71      0.79      3000

avg / total       0.83      0.81      0.81      6000

Train SubjectB resampled_duplication
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 8.638601779937744
[[2704  296]
```

```
 [ 445 2555]]
             precision    recall  f1-score   support

          0       0.86      0.90      0.88      3000
          1       0.90      0.85      0.87      3000

avg / total       0.88      0.88      0.88      6000
```

5. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 5:

```
Train SubjectA resampled_borderline pca 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 10.780130386352539
[[2715  285]
 [ 692 2308]]
             precision    recall  f1-score   support

          0       0.80      0.91      0.85      3000
          1       0.89      0.77      0.83      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_borderline pca 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 10.062365770339966
[[2736  264]
 [ 719 2281]]
             precision    recall  f1-score   support

          0       0.79      0.91      0.85      3000
          1       0.90      0.76      0.82      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_borderline pca 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 12.073383569717407
[[2703  297]
 [ 667 2333]]
```

```
             precision    recall  f1-score   support

          0       0.80      0.90      0.85      3000
          1       0.89      0.78      0.83      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_borderline pca 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 13.091113567352295
[[2641  359]
 [ 614 2386]]
             precision    recall  f1-score   support

          0       0.81      0.88      0.84      3000
          1       0.87      0.80      0.83      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_borderline pca 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 15.2460777759552
[[2582  418]
 [ 498 2502]]
             precision    recall  f1-score   support

          0       0.84      0.86      0.85      3000
          1       0.86      0.83      0.85      3000

avg / total       0.85      0.85      0.85      6000

Train SubjectA resampled_borderline pca 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 15.778558492660522
[[2535  465]
 [ 390 2610]]
             precision    recall  f1-score   support

          0       0.87      0.84      0.86      3000
          1       0.85      0.87      0.86      3000
```

```
avg / total       0.86      0.86      0.86      6000

Train SubjectA resampled_borderline pca 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 16.872251987457275
[[2521  479]
 [ 488 2512]]
             precision    recall  f1-score   support

          0       0.84      0.84      0.84      3000
          1       0.84      0.84      0.84      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectB resampled_borderline pca 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 6.597111940383911
[[2700  300]
 [ 288 2712]]
             precision    recall  f1-score   support

          0       0.90      0.90      0.90      3000
          1       0.90      0.90      0.90      3000

avg / total       0.90      0.90      0.90      6000

Train SubjectB resampled_borderline pca 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 5.328200578689575
[[2681  319]
 [ 274 2726]]
             precision    recall  f1-score   support

          0       0.91      0.89      0.90      3000
          1       0.90      0.91      0.90      3000

avg / total       0.90      0.90      0.90      6000

Train SubjectB resampled_borderline pca 33 channel
target test data =  3000
non target test data =  3000
```

```
target train data =  2550
non target train data =  2550
Training time: 5.2175633907318115
[[2659  341]
 [ 341 2659]]
             precision    recall  f1-score   support

          0       0.89      0.89      0.89      3000
          1       0.89      0.89      0.89      3000

avg / total       0.89      0.89      0.89      6000

Train SubjectB resampled_borderline pca 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 7.28589391708374
[[2626  374]
 [ 393 2607]]
             precision    recall  f1-score   support

          0       0.87      0.88      0.87      3000
          1       0.87      0.87      0.87      3000

avg / total       0.87      0.87      0.87      6000

Train SubjectB resampled_borderline pca 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 9.671019554138184
[[2552  448]
 [ 374 2626]]
             precision    recall  f1-score   support

          0       0.87      0.85      0.86      3000
          1       0.85      0.88      0.86      3000

avg / total       0.86      0.86      0.86      6000

Train SubjectB resampled_borderline pca 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 9.872426271438599
[[2471  529]
 [ 378 2622]]
```

```
             precision    recall  f1-score   support

          0       0.87      0.82      0.84      3000
          1       0.83      0.87      0.85      3000

avg / total       0.85      0.85      0.85      6000

Train SubjectB resampled_borderline pca 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 12.10817551612854
[[2383  617]
 [ 458 2542]]
             precision    recall  f1-score   support

          0       0.84      0.79      0.82      3000
          1       0.80      0.85      0.83      3000

avg / total       0.82      0.82      0.82      6000

Train SubjectA resampled_adasyn pca 46 channel
target test data =  2948
non target test data =  3000
target train data =  2590
non target train data =  2550
Training time: 50.19397830963135
[[2329  671]
 [1184 1764]]
             precision    recall  f1-score   support

          0       0.66      0.78      0.72      3000
          1       0.72      0.60      0.66      2948

avg / total       0.69      0.69      0.69      5948

Train SubjectA resampled_adasyn pca 40 channel
target test data =  2935
non target test data =  3000
target train data =  2508
non target train data =  2550
Training time: 58.4102349281311
[[2300  700]
 [1143 1792]]
             precision    recall  f1-score   support

          0       0.67      0.77      0.71      3000
          1       0.72      0.61      0.66      2935
```

```
avg / total       0.69      0.69      0.69      5935

Train SubjectA resampled_adasyn pca 33 channel
target test data =  2928
non target test data =  3000
target train data =  2606
non target train data =  2550
Training time: 57.0809428691864
[[2268  732]
 [1009 1919]]
             precision    recall  f1-score   support

          0       0.69      0.76      0.72      3000
          1       0.72      0.66      0.69      2928

avg / total       0.71      0.71      0.71      5928

Train SubjectA resampled_adasyn pca 26 channel
target test data =  2935
non target test data =  3000
target train data =  2485
non target train data =  2550
Training time: 76.28229236602783
[[2229  771]
 [ 935 2000]]
             precision    recall  f1-score   support

          0       0.70      0.74      0.72      3000
          1       0.72      0.68      0.70      2935

avg / total       0.71      0.71      0.71      5935

Train SubjectA resampled_adasyn pca 19 channel
target test data =  2924
non target test data =  3000
target train data =  2538
non target train data =  2550
Training time: 51.849284410476685
[[2222  778]
 [ 925 1999]]
             precision    recall  f1-score   support

          0       0.71      0.74      0.72      3000
          1       0.72      0.68      0.70      2924

avg / total       0.71      0.71      0.71      5924

Train SubjectA resampled_adasyn pca 13 channel
target test data =  2916
non target test data =  3000
```

```
target train data =  2532
non target train data =  2550
Training time: 34.697101354599
[[2203  797]
 [ 845 2071]]
             precision    recall  f1-score   support

          0       0.72      0.73      0.73      3000
          1       0.72      0.71      0.72      2916

avg / total       0.72      0.72      0.72      5916

Train SubjectA resampled_adasyn pca 8 channel
target test data =  2913
non target test data =  3000
target train data =  2506
non target train data =  2550
Training time: 30.148164749145508
[[2193  807]
 [ 901 2012]]
             precision    recall  f1-score   support

          0       0.71      0.73      0.72      3000
          1       0.71      0.69      0.70      2913

avg / total       0.71      0.71      0.71      5913

Train SubjectB resampled_adasyn pca 46 channel
target test data =  2897
non target test data =  3000
target train data =  2602
non target train data =  2550
Training time: 39.664607524871826
[[2295  705]
 [ 862 2035]]
             precision    recall  f1-score   support

          0       0.73      0.77      0.75      3000
          1       0.74      0.70      0.72      2897

avg / total       0.73      0.73      0.73      5897

Train SubjectB resampled_adasyn pca 40 channel
target test data =  2893
non target test data =  3000
target train data =  2527
non target train data =  2550
Training time: 37.31762623786926
[[2318  682]
 [ 800 2093]]
```

```
             precision    recall  f1-score   support

          0       0.74      0.77      0.76      3000
          1       0.75      0.72      0.74      2893

avg / total       0.75      0.75      0.75      5893

Train SubjectB resampled_adasyn pca 33 channel
target test data =  2890
non target test data =  3000
target train data =  2537
non target train data =  2550
Training time: 40.28728413581848
[[2291  709]
 [ 778 2112]]
             precision    recall  f1-score   support

          0       0.75      0.76      0.75      3000
          1       0.75      0.73      0.74      2890

avg / total       0.75      0.75      0.75      5890

Train SubjectB resampled_adasyn pca 26 channel
target test data =  2890
non target test data =  3000
target train data =  2626
non target train data =  2550
Training time: 34.46021628379822
[[2222  778]
 [ 734 2156]]
             precision    recall  f1-score   support

          0       0.75      0.74      0.75      3000
          1       0.73      0.75      0.74      2890

avg / total       0.74      0.74      0.74      5890

Train SubjectB resampled_adasyn pca 19 channel
target test data =  2903
non target test data =  3000
target train data =  2558
non target train data =  2550
Training time: 36.646196603775024
[[2234  766]
 [ 760 2143]]
             precision    recall  f1-score   support

          0       0.75      0.74      0.75      3000
          1       0.74      0.74      0.74      2903
```

```
avg / total       0.74      0.74      0.74      5903

Train SubjectB resampled_adasyn pca 13 channel
target test data =  2926
non target test data =  3000
target train data =  2539
non target train data =  2550
Training time: 31.182438611984253
[[2185  815]
 [ 762 2164]]
             precision    recall  f1-score   support

          0       0.74      0.73      0.73      3000
          1       0.73      0.74      0.73      2926

avg / total       0.73      0.73      0.73      5926

Train SubjectB resampled_adasyn pca 8 channel
target test data =  2950
non target test data =  3000
target train data =  2548
non target train data =  2550
Training time: 29.652641534805298
[[2100  900]
 [ 829 2121]]
             precision    recall  f1-score   support

          0       0.72      0.70      0.71      3000
          1       0.70      0.72      0.71      2950

avg / total       0.71      0.71      0.71      5950

Train SubjectA resampled_duplication pca 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 13.074543714523315
[[2715  285]
 [ 835 2165]]
             precision    recall  f1-score   support

          0       0.76      0.91      0.83      3000
          1       0.88      0.72      0.79      3000

avg / total       0.82      0.81      0.81      6000

Train SubjectA resampled_duplication pca 40 channel
target test data =  3000
non target test data =  3000
```

```
target train data =  2550
non target train data =  2550
Training time: 11.467800378799438
[[2703  297]
 [ 835 2165]]
             precision    recall  f1-score   support

          0       0.76      0.90      0.83      3000
          1       0.88      0.72      0.79      3000

avg / total       0.82      0.81      0.81      6000

Train SubjectA resampled_duplication pca 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 16.82464909553528
[[2668  332]
 [ 800 2200]]
             precision    recall  f1-score   support

          0       0.77      0.89      0.82      3000
          1       0.87      0.73      0.80      3000

avg / total       0.82      0.81      0.81      6000

Train SubjectA resampled_duplication pca 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 23.12406849861145
[[2599  401]
 [ 745 2255]]
             precision    recall  f1-score   support

          0       0.78      0.87      0.82      3000
          1       0.85      0.75      0.80      3000

avg / total       0.81      0.81      0.81      6000

Train SubjectA resampled_duplication pca 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 29.24226951599121
[[2538  462]
 [ 605 2395]]
```

```
             precision    recall  f1-score   support

          0       0.81      0.85      0.83      3000
          1       0.84      0.80      0.82      3000

avg / total       0.82      0.82      0.82      6000

Train SubjectA resampled_duplication pca 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 31.49385118484497
[[2521  479]
 [ 565 2435]]
             precision    recall  f1-score   support

          0       0.82      0.84      0.83      3000
          1       0.84      0.81      0.82      3000

avg / total       0.83      0.83      0.83      6000

Train SubjectA resampled_duplication pca 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 29.05584740638733
[[2473  527]
 [ 600 2400]]
             precision    recall  f1-score   support

          0       0.80      0.82      0.81      3000
          1       0.82      0.80      0.81      3000

avg / total       0.81      0.81      0.81      6000

Train SubjectB resampled_duplication pca 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 8.87480354309082
[[2700  300]
 [ 445 2555]]
             precision    recall  f1-score   support

          0       0.86      0.90      0.88      3000
          1       0.89      0.85      0.87      3000
```

```
avg / total       0.88      0.88      0.88      6000

Train SubjectB resampled_duplication pca 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 7.128588914871216
[[2681  319]
 [ 460 2540]]
             precision    recall  f1-score   support

          0       0.85      0.89      0.87      3000
          1       0.89      0.85      0.87      3000

avg / total       0.87      0.87      0.87      6000

Train SubjectB resampled_duplication pca 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 8.32839059829712
[[2658  342]
 [ 475 2525]]
             precision    recall  f1-score   support

          0       0.85      0.89      0.87      3000
          1       0.88      0.84      0.86      3000

avg / total       0.86      0.86      0.86      6000

Train SubjectB resampled_duplication pca 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 12.784918546676636
[[2611  389]
 [ 565 2435]]
             precision    recall  f1-score   support

          0       0.82      0.87      0.85      3000
          1       0.86      0.81      0.84      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectB resampled_duplication pca 19 channel
target test data =  3000
non target test data =  3000
```

```
target train data =  2550
non target train data =  2550
Training time: 24.229387283325195
[[2479  521]
 [ 500 2500]]
             precision    recall  f1-score   support

          0       0.83      0.83      0.83      3000
          1       0.83      0.83      0.83      3000

avg / total       0.83      0.83      0.83      6000

Train SubjectB resampled_duplication pca 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 35.55623483657837
[[2399  601]
 [ 450 2550]]
             precision    recall  f1-score   support

          0       0.84      0.80      0.82      3000
          1       0.81      0.85      0.83      3000

avg / total       0.83      0.82      0.82      6000

Train SubjectB resampled_duplication pca 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 29.745059490203857
[[2363  637]
 [ 580 2420]]
             precision    recall  f1-score   support

          0       0.80      0.79      0.80      3000
          1       0.79      0.81      0.80      3000

avg / total       0.80      0.80      0.80      6000
```

6. Keluaran dari perangkat lunak untuk proses klasifikasi pada skenario uji coba 6:

```
Train SubjectA resampled_borderline lda 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
```

```
non target train data =  2550
Training time: 0.15622234344482422
[[2169  831]
 [ 272 2728]]
             precision    recall  f1-score   support

          0       0.89      0.72      0.80      3000
          1       0.77      0.91      0.83      3000

avg / total       0.83      0.82      0.81      6000

Train SubjectA resampled_borderline lda 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.16234850883483887
[[2272  728]
 [ 339 2661]]
             precision    recall  f1-score   support

          0       0.87      0.76      0.81      3000
          1       0.79      0.89      0.83      3000

avg / total       0.83      0.82      0.82      6000

Train SubjectA resampled_borderline lda 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.15621519088745117
[[2338  662]
 [ 330 2670]]
             precision    recall  f1-score   support

          0       0.88      0.78      0.82      3000
          1       0.80      0.89      0.84      3000

avg / total       0.84      0.83      0.83      6000

Train SubjectA resampled_borderline lda 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.19299888610839844
[[2289  711]
 [ 406 2594]]
             precision    recall  f1-score   support
```

```
          0       0.85      0.76      0.80      3000
          1       0.78      0.86      0.82      3000

avg / total       0.82      0.81      0.81      6000

Train SubjectA resampled_borderline lda 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.1965188980102539
[[2250  750]
 [ 333 2667]]
             precision    recall  f1-score   support

          0       0.87      0.75      0.81      3000
          1       0.78      0.89      0.83      3000

avg / total       0.83      0.82      0.82      6000

Train SubjectA resampled_borderline lda 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.21869587898254395
[[2216  784]
 [ 281 2719]]
             precision    recall  f1-score   support

          0       0.89      0.74      0.81      3000
          1       0.78      0.91      0.84      3000

avg / total       0.83      0.82      0.82      6000

Train SubjectA resampled_borderline lda 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.2215421199798584
[[2257  743]
 [ 394 2606]]
             precision    recall  f1-score   support

          0       0.85      0.75      0.80      3000
          1       0.78      0.87      0.82      3000

avg / total       0.81      0.81      0.81      6000
```

```
Train SubjectB resampled_borderline lda 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.12496781349182129
[[2390  610]
 [  62 2938]]
             precision    recall  f1-score   support

          0       0.97      0.80      0.88      3000
          1       0.83      0.98      0.90      3000

avg / total       0.90      0.89      0.89      6000

Train SubjectB resampled_borderline lda 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.12344646453857422
[[2341  659]
 [ 137 2863]]
             precision    recall  f1-score   support

          0       0.94      0.78      0.85      3000
          1       0.81      0.95      0.88      3000

avg / total       0.88      0.87      0.87      6000

Train SubjectB resampled_borderline lda 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.15621376037597656
[[2284  716]
 [ 104 2896]]
             precision    recall  f1-score   support

          0       0.96      0.76      0.85      3000
          1       0.80      0.97      0.88      3000

avg / total       0.88      0.86      0.86      6000

Train SubjectB resampled_borderline lda 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
```

```
non target train data =  2550
Training time: 0.18748164176940918
[[2245  755]
 [ 194 2806]]
             precision    recall  f1-score   support

          0       0.92      0.75      0.83      3000
          1       0.79      0.94      0.86      3000

avg / total       0.85      0.84      0.84      6000

Train SubjectB resampled_borderline lda 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.17990374565124512
[[2290  710]
 [ 194 2806]]
             precision    recall  f1-score   support

          0       0.92      0.76      0.84      3000
          1       0.80      0.94      0.86      3000

avg / total       0.86      0.85      0.85      6000

Train SubjectB resampled_borderline lda 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.18744921684265137
[[2208  792]
 [ 223 2777]]
             precision    recall  f1-score   support

          0       0.91      0.74      0.81      3000
          1       0.78      0.93      0.85      3000

avg / total       0.84      0.83      0.83      6000

Train SubjectB resampled_borderline lda 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.250011682510376
[[2097  903]
 [ 241 2759]]
             precision    recall  f1-score   support
```

```
          0       0.90      0.70      0.79      3000
          1       0.75      0.92      0.83      3000

avg / total       0.83      0.81      0.81      6000

Train SubjectA resampled_adasyn lda 46 channel
target test data =  2988
non target test data =  3000
target train data =  2537
non target train data =  2550
Training time: 0.3020360469818115
[[1914 1086]
 [ 705 2283]]
             precision    recall  f1-score   support

          0       0.73      0.64      0.68      3000
          1       0.68      0.76      0.72      2988

avg / total       0.70      0.70      0.70      5988

Train SubjectA resampled_adasyn lda 40 channel
target test data =  3071
non target test data =  3000
target train data =  2547
non target train data =  2550
Training time: 0.3148660659790039
[[1964 1036]
 [ 737 2334]]
             precision    recall  f1-score   support

          0       0.73      0.65      0.69      3000
          1       0.69      0.76      0.72      3071

avg / total       0.71      0.71      0.71      6071

Train SubjectA resampled_adasyn lda 33 channel
target test data =  2991
non target test data =  3000
target train data =  2555
non target train data =  2550
Training time: 0.3120455741882324
[[2001  999]
 [ 820 2171]]
             precision    recall  f1-score   support

          0       0.71      0.67      0.69      3000
          1       0.68      0.73      0.70      2991

avg / total       0.70      0.70      0.70      5991
```

```
Train SubjectA resampled_adasyn lda 26 channel
target test data =  3022
non target test data =  3000
target train data =  2564
non target train data =  2550
Training time: 0.32616734504699707
[[1985 1015]
 [ 746 2276]]
             precision    recall  f1-score   support

          0       0.73      0.66      0.69      3000
          1       0.69      0.75      0.72      3022

avg / total       0.71      0.71      0.71      6022

Train SubjectA resampled_adasyn lda 19 channel
target test data =  2988
non target test data =  3000
target train data =  2575
non target train data =  2550
Training time: 0.4066591262817383
[[1976 1024]
 [ 823 2165]]
             precision    recall  f1-score   support

          0       0.71      0.66      0.68      3000
          1       0.68      0.72      0.70      2988

avg / total       0.69      0.69      0.69      5988

Train SubjectA resampled_adasyn lda 13 channel
target test data =  3026
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.375063419342041
[[1958 1042]
 [ 877 2149]]
             precision    recall  f1-score   support

          0       0.69      0.65      0.67      3000
          1       0.67      0.71      0.69      3026

avg / total       0.68      0.68      0.68      6026

Train SubjectA resampled_adasyn lda 8 channel
target test data =  2999
non target test data =  3000
target train data =  2538
```

```
non target train data =  2550
Training time: 0.40568971633911133
[[2034  966]
 [ 927 2072]]
             precision    recall  f1-score   support

          0       0.69      0.68      0.68      3000
          1       0.68      0.69      0.69      2999

avg / total       0.68      0.68      0.68      5999

Train SubjectB resampled_adasyn lda 46 channel
target test data =  2989
non target test data =  3000
target train data =  2543
non target train data =  2550
Training time: 0.2501087188720703
[[2006  994]
 [ 723 2266]]
             precision    recall  f1-score   support

          0       0.74      0.67      0.70      3000
          1       0.70      0.76      0.73      2989

avg / total       0.72      0.71      0.71      5989

Train SubjectB resampled_adasyn lda 40 channel
target test data =  2954
non target test data =  3000
target train data =  2538
non target train data =  2550
Training time: 0.28658199310302734
[[2003  997]
 [ 745 2209]]
             precision    recall  f1-score   support

          0       0.73      0.67      0.70      3000
          1       0.69      0.75      0.72      2954

avg / total       0.71      0.71      0.71      5954

Train SubjectB resampled_adasyn lda 33 channel
target test data =  2959
non target test data =  3000
target train data =  2541
non target train data =  2550
Training time: 0.31242895126342773
[[1992 1008]
 [ 698 2261]]
             precision    recall  f1-score   support
```

```
          0       0.74      0.66      0.70      3000
          1       0.69      0.76      0.73      2959

avg / total       0.72      0.71      0.71      5959

Train SubjectB resampled_adasyn lda 26 channel
target test data =  3012
non target test data =  3000
target train data =  2566
non target train data =  2550
Training time: 0.32152366638183594
[[2021  979]
 [ 787 2225]]
             precision    recall  f1-score   support

          0       0.72      0.67      0.70      3000
          1       0.69      0.74      0.72      3012

avg / total       0.71      0.71      0.71      6012

Train SubjectB resampled_adasyn lda 19 channel
target test data =  3036
non target test data =  3000
target train data =  2513
non target train data =  2550
Training time: 0.4357905387878418
[[2006  994]
 [ 822 2214]]
             precision    recall  f1-score   support

          0       0.71      0.67      0.69      3000
          1       0.69      0.73      0.71      3036

avg / total       0.70      0.70      0.70      6036

Train SubjectB resampled_adasyn lda 13 channel
target test data =  3034
non target test data =  3000
target train data =  2563
non target train data =  2550
Training time: 0.46398329734802246
[[1963 1037]
 [ 827 2207]]
             precision    recall  f1-score   support

          0       0.70      0.65      0.68      3000
          1       0.68      0.73      0.70      3034

avg / total       0.69      0.69      0.69      6034
```

```
Train SubjectB resampled_adasyn lda 8 channel
target test data =  3044
non target test data =  3000
target train data =  2535
non target train data =  2550
Training time: 0.3928866386413574
[[1944 1056]
 [ 867 2177]]
             precision    recall  f1-score   support

          0       0.69      0.65      0.67      3000
          1       0.67      0.72      0.69      3044

avg / total       0.68      0.68      0.68      6044

Train SubjectA resampled_duplication lda 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.031286001205444336
[[2592  408]
 [ 665 2335]]
             precision    recall  f1-score   support

          0       0.80      0.86      0.83      3000
          1       0.85      0.78      0.81      3000

avg / total       0.82      0.82      0.82      6000

Train SubjectA resampled_duplication lda 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.015618562698364258
[[2610  390]
 [ 605 2395]]
             precision    recall  f1-score   support

          0       0.81      0.87      0.84      3000
          1       0.86      0.80      0.83      3000

avg / total       0.84      0.83      0.83      6000

Train SubjectA resampled_duplication lda 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
```

```
non target train data =  2550
Training time: 0.0369420051574707
[[2554  446]
 [ 525 2475]]
             precision    recall  f1-score   support

          0       0.83      0.85      0.84      3000
          1       0.85      0.82      0.84      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_duplication lda 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.04685354232788086
[[2563  437]
 [ 500 2500]]
             precision    recall  f1-score   support

          0       0.84      0.85      0.85      3000
          1       0.85      0.83      0.84      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_duplication lda 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.04682302474975586
[[2537  463]
 [ 520 2480]]
             precision    recall  f1-score   support

          0       0.83      0.85      0.84      3000
          1       0.84      0.83      0.83      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_duplication lda 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.07813453674316406
[[2525  475]
 [ 490 2510]]
             precision    recall  f1-score   support
```

```
          0       0.84      0.84      0.84      3000
          1       0.84      0.84      0.84      3000

avg / total       0.84      0.84      0.84      6000

Train SubjectA resampled_duplication lda 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.14064717292785645
[[2480  520]
 [ 560 2440]]
             precision    recall  f1-score   support

          0       0.82      0.83      0.82      3000
          1       0.82      0.81      0.82      3000

avg / total       0.82      0.82      0.82      6000

Train SubjectB resampled_duplication lda 46 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.015659332275390625
[[2520  480]
 [ 245 2755]]
             precision    recall  f1-score   support

          0       0.91      0.84      0.87      3000
          1       0.85      0.92      0.88      3000

avg / total       0.88      0.88      0.88      6000

Train SubjectB resampled_duplication lda 40 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.015663862228393555
[[2560  440]
 [ 250 2750]]
             precision    recall  f1-score   support

          0       0.91      0.85      0.88      3000
          1       0.86      0.92      0.89      3000

avg / total       0.89      0.89      0.88      6000
```

```
Train SubjectB resampled_duplication lda 33 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.031803131103515625
[[2606  394]
 [ 300 2700]]
             precision    recall  f1-score   support

          0       0.90      0.87      0.88      3000
          1       0.87      0.90      0.89      3000

avg / total       0.88      0.88      0.88      6000

Train SubjectB resampled_duplication lda 26 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.03122425079345703
[[2560  440]
 [ 325 2675]]
             precision    recall  f1-score   support

          0       0.89      0.85      0.87      3000
          1       0.86      0.89      0.87      3000

avg / total       0.87      0.87      0.87      6000

Train SubjectB resampled_duplication lda 19 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.06252408027648926
[[2466  534]
 [ 350 2650]]
             precision    recall  f1-score   support

          0       0.88      0.82      0.85      3000
          1       0.83      0.88      0.86      3000

avg / total       0.85      0.85      0.85      6000

Train SubjectB resampled_duplication lda 13 channel
target test data =  3000
non target test data =  3000
target train data =  2550
```

```
non target train data =  2550
Training time: 0.08080554008483887
[[2459  541]
 [ 365 2635]]
             precision    recall  f1-score   support

          0       0.87      0.82      0.84      3000
          1       0.83      0.88      0.85      3000

avg / total       0.85      0.85      0.85      6000

Train SubjectB resampled_duplication lda 8 channel
target test data =  3000
non target test data =  3000
target train data =  2550
non target train data =  2550
Training time: 0.1370232105255127
[[2385  615]
 [ 495 2505]]
             precision    recall  f1-score   support

          0       0.83      0.80      0.81      3000
          1       0.80      0.83      0.82      3000

avg / total       0.82      0.81      0.81      6000
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Vincentius dilahirkan di Jakarta pada tanggal 14 Desember 1997 dan dibesarkan di Kota Bekasi. Sejak usia sekolah, penulis sudah memiliki ketertarikan dalam bidang teknologi informasi.

Penulis menempuh pendidikan di SD Trinitas (2003-2009), SMP Don Bosco III Cikarang (2009-2012), dan SMA Don Bosco III Cikarang (2012-2015). Setelah lulus SMA penulis melanjutkan ke jenjang pendidikan tinggi di Departemen Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Rumpun Mata Kuliah dominan yang diambil oleh penulis pada saat menempuh pendidikan di Departemen Informatika ITS adalah Komputasi Cerdas dan Visi.

Selama menempuh kuliah penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer (HMTC) ITS pada Departemen Kewirausahaan di tahun 2016-2017. Selain itu, penulis juga mengabdi sebagai asisten pengajar mata kuliah Dasar Pemrograman pada Semester Gasal tahun ajaran 2017-2018.